OPTIMIZATION OF A COMPUTERIZED RESPIRATORY MEASUREMENT SYSTEM

by

GARY IRWIN NOYES

B.S., Kansas State University, 1985

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1987

Approved by:

Richard R. Gallagher
Major Professor

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF TERMS

ASCII	American Standard Code for Information Interchange. This is a standard method of representing all printable characters in the English language.
bit	The smallest unit of computer memory. Only two states can be represented, either 1 or 0.
BTPS	Body Temperature and Pressure; Saturated
byte	A unit of computer memory. One byte is enough memory to store one ASCII character.
DAM	Data Acquisition Module. The analog-to-digital converter system used to collect data.
FRC	Functional Residual Capacity. This term represents the resting, end-expiratory lung volume of a subject.
GPIB	General Purpose Interface Bus. A computer interface developed by Hewlett-Packard that is used for connecting a computer to peripherals. (e.g., disk drives, printers, and plotters are commonly interfaced using this bus.)
GPIO	General Purpose Input and Output interface. This is a Hewlett-Packard interface that is typically used to interface non-standard peripherals to a host computer.
HP	An abbreviation of the company name Hewlett-Packard.

IEEE	Institute of Electrical and Electronics Engineers.
IEEE-488	Another name for the GPIB which is described above.
I/O	Input and/or output. This term is commonly used when referring to computer peripherals that are used for data input and/or output.
Kbyte	1024 bytes of computer memory.
LED	Light Emitting Diode.
Mbyte	1024 Kbytes of computer memory.
PC	Personal Computer.
PTM	Pneumotachometer. A device that is used to measure gas flow.
RAM	Random Access Memory. This term represents the portion of computer memory that can be both written to and read from.
RS-232C	A standard serial interface that can be used to interface printers to computers.
run-time errors	A computer or program error that occurs during the execution of a computer program. (e.g., divide by zero is a common run-time error.
Serial interface	A computer interface that sends each only one bit of information at a time.
STPD	Standard Temperature and Pressure; Dry

I. INTRODUCTION

A computerized breath-by-breath respiratory measurement system has been developed in the bioengineering research laboratory at Kansas State University. The project began as a study of the respiratory systems of calves exercising on a treadmill (1). The measurement system was later modified to accommodate human subjects exercising on a bicycle ergometer (2,3).

The term "breath-by-breath" is used because the measurement system is capable of measuring oxygen consumption and carbon dioxide production on a per breath basis. The approach can be compared with conventional methods such as the Douglas bag technique. The Douglas bag technique uses a meterological balloon to collect an exercising subject's expirate. The volumes of oxygen and carbon dioxide present in the collected gas are used to determine the average rate of oxygen consumption and carbon dioxide production during the interval of time of gas collection. The bag method is able to evaluate only the steady-state response.

For the purpose of this thesis, a steady-state of exercise is defined as the time period after all

transients associated with the start of exercise have diminished. With the Douglas bag technique, researchers are unable to determine when a subject reaches steady state. Breath-by-breath techniques eliminate this unknown by allowing the researcher to view the on- and off-transient curves caused by starting and stopping exercise. By viewing the breath-by-breath data, researchers can identify when subjects reach a steady- state condition. For example, the interval from 216 to 378 seconds in Figure 1.1 represents a typical steady-state of alveolar carbon dioxide production during exercise. All of the breaths occurring during the steady-state period can be analyzed, effectively emulating the Douglas bag method.

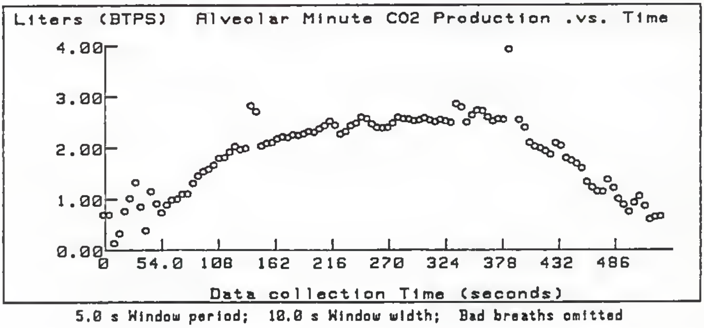


Figure 1.1. Example of steady-state alveolar carbon dioxide production during exercise.

To facilitate future analysis of collected respiratory data, all of the data are stored in disk files.

Because the data are permanently stored, the analysis techniques can be altered without collecting new data. The data can be analyzed by selecting different intervals of time over which to average data. These time intervals are defined as time windows, and a typical time window might include the interval of steady-state exercise.

The analysis algorithms of the breath-by-breath system were originally implemented using a hybrid of Hewlett-Packard (HP) BASIC and HP Pascal operating systems and languages. The BASIC language was used for data analysis because it is easy to use. The Pascal language was used for data collection because it is fast and can support assembly language programs. Two assembly language programs were required to control the Data Acquisition Module (DAM).

Understanding both the BASIC and Pascal operating systems is difficult. Users must be versed in the operation of both systems, the entire set of programs, and the storage conventions for efficient operation of the hybrid systems.

This thesis presents enhancements to the operation of the computerized respiratory measurement system. The current version of the analysis software is written in Pascal and is referred to as Version 3. Version 2 represents the previous implementation of the analysis software (3).

II. RESEARCH OBJECTIVES

The primary goal of this research is to modify the respiratory measurement system so that it has fast execution times and is easy to use. If operation of the system is straightforward then researchers from other disciplines can incorporate the system into their respiratory physiology studies. The goals of this research are as follows:

1. Select one operating system,
2. Simplify the operating procedure,
3. Increase the execution speed,
4. Improve the measurement accuracy,
5. Produce code that is portable to other systems,
6. Provide capability for longer data runs.

The implementation and verification of these goals are presented in Chapters IV and V. Chapter III presents a description of the hardware and software aspects of the respiratory measurement system.

III. DESCRIPTION OF THE RESPIRATORY MEASUREMENT SYSTEM

The computerized respiratory measurement system is used to calculate volumes of oxygen consumption and carbon dioxide production on a per breath basis. Averaging techniques can be used to combine a time window of breaths, simulating the results obtained by using standard techniques such as the collection of expirate gasses with a Douglas bag. A block diagram of the computerized system is shown in Figure 2.1. A bicycle ergometer provides a controlled workload to exercise the subjects. The bicycle ergometer provides an equivalent workload regardless of a subject's level of training (4).

Calculating the respiratory parameters requires the monitoring of several physiological signals. Four of the physiological signals are monitored by the computerized instrumentation system. Other signals, such as barometric pressure, are only measured once and assumed to remain constant through completion of the testing protocol. A data acquisition module converts voltages produced by the transducers into binary data that are readable by the computer. The transducers used in the respiratory measurement system are described in Section 3.1.

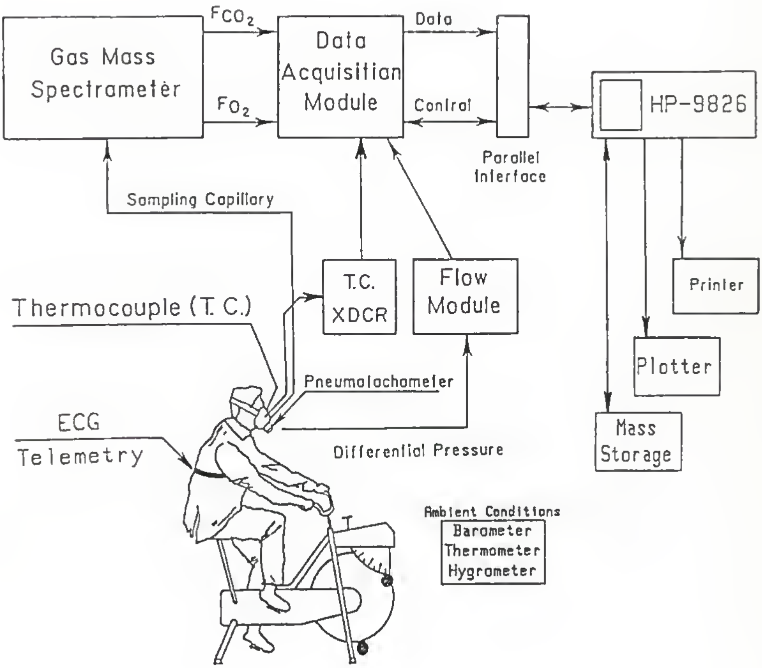


Figure 2.1. Block diagram of the computerized respiratory measurement system. Reproduced from Masters (3).

3.1 Transducers - Steady-State

Several measurements are made only once and assumed to remain constant during an exercise run. The barometric pressure, relative humidity, and room temperature are measured prior to a run using a conventional barometer, hygrometer, and thermometer. These ambient conditions are required by the software algorithms to convert gas volumes to STPD (standard temperature and pressure; dry) conditions.

The subject's age, weight, and height are also required. This information is used to approximate the subject's functional residual capacity (FRC). Knowledge of the FRC allows gas volume calculations to be made with respect to the alveolar region of the lung rather than with respect to volumes measured at the mouth (5). The subject's end-exercise body temperature is also measured. The body temperature is used for converting gas volumes from STPD to BTPS (body temperature and pressure; saturated) conditions.

3.2 Transducers - Continuous

Five signals are continuously monitored by the respiratory measurement system. Four of the signals are monitored by the computerized system and the fifth, heart rate, is monitored by a telemetry system. The four respiratory signals monitored by the computer are instan-

taneous concentrations of carbon dioxide, oxygen, gas flow rate, and gas temperature. Section 3.2.5 describes the placement of the transducers. The calibration procedure for the continuous signal transducers is described in Appendix A, Operators Guide to the Respiratory Measurement System.

3.2.1 Gas Mass Spectrometer

A rapidly responding Perkin-Elmer model 1100 mass spectrometer (GMS) is used to measure gas concentrations. Although there are several gases present in the respiratory flow, only carbon dioxide and oxygen are monitored by the computer. It is assumed that a fixed concentration (0.50%) of inert gases are present. The subject's expiratory gas is assumed to be saturated with water vapor. The vapor concentration of the inspiratory gas is assumed to be equivalent to ambient conditions. The balance of the gas volume is assumed to be nitrogen.

A significant amount of time is required for the GMS to sample and analyze gas concentrations. The GMS time delay varies from 300 to 700 milliseconds, depending on the length and condition of the sampling capillary. An algorithm has been developed that time-aligns the gas concentration signals with the flow signal (2). The time delay is calculated on a per breath basis, allowing the program to respond to changes in the sampling capillary's

flow characteristics.

3.2.2 Thermocouple

Gas temperature is measured using a 0.001 inch diameter type E (chromel-constantan), bead welded, thermocouple (3). Response time of the thermocouple is 4.5 ms. The thermocouple probe is imbedded in an epoxy-resin base for strength, with only the tip exposed to the respiratory gasses.

3.2.3 Gas Flow

Flow measurements are obtained using a model 3813 Rudolph pneumotachometer (PTM) and a Godart differential pressure transducer. Figure 3.1 shows an illustration of the PTM. The outer shell of the PTM is heated to reduce condensation on the internal screens. As air passes through the three screens a differential pressure is produced.

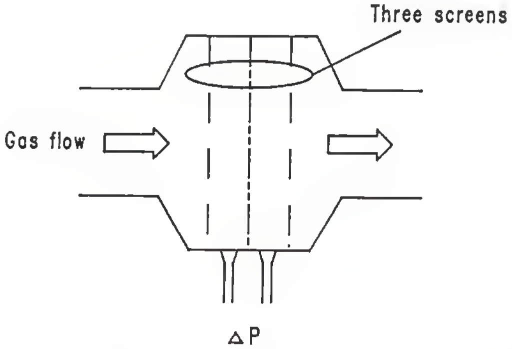


Figure 3.1. Schematic diagram of the Rudolph pneumotachometer.

Although specifications indicate a linear response from zero to sixteen liters per second for the PTM (6), Section 4.4.2 shows that the PTM and Godart box combination is not linear for flows less than one liter per second. Flows greater than four liters per second have not been thoroughly evaluated.

3.2.4 Heart Rate

Heart rate is monitored using an Amerec 150 Sport Tester PE 2000 commercial telemetry system manufactured by Polar Electro. The subject wears an electrode belt strapped around the chest. A small transmitter attached to the electrode belt provides a range of nearly three feet. A receiver is either worn on the subject's wrist or attached to the handle bars of the bicycle. During exercise, the receiver displays the current heart rate after every fifth beat. The heart rate is also averaged on thirty second intervals and stored in the receiver's memory.

3.2.5 Face Mask

The GMS sampling capillary, PTM, and thermocouple must be placed near the subject's mouth. A face mask is used to mount the transducers. The face mask is chosen over a mouth piece for subject convenience, allowing the subject to breathe using the mouth, nose, or both. Figure 3.2 shows how the transducers are connected to the face

mask. An attempt is made to minimize added dead space.

The order of the sensor placement is important. The thermocouple is placed as close to the mouth as possible in order to measure the expirate temperature before it cools. The GMS sampling capillary is placed near the exit of the mask assembly. A fan blowing across the subject's face ensures that there is not a buildup of expirate, which has a high concentration of carbon dioxide, near the exit of the mask. The PTM is located between the thermocouple and the GMS sampling capillary.

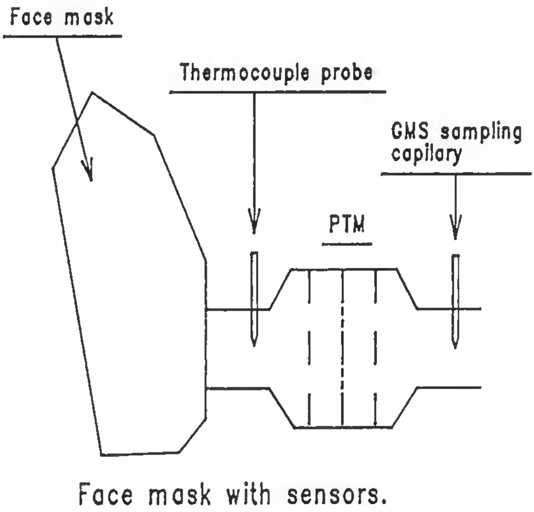


Figure 3.2. Block diagram illustrating the transducer connections to the face mask.

3.3 Data Acquisition Module

The four analog voltages produced by the GMS, flow, and temperature transducers must be converted into computer readable form. A custom-made DAM (1,2,7), is used for the analog-to-digital conversions. Eight, 12-bit channels are available; however, only four channels are implemented. All four channels are sampled simultaneously and one 12-bit successive approximation analog-to-digital converter is used to sequentially measure the voltages. A programmable timer is also provided by the DAM, which can be used to control the sampling rate. The DAM is interfaced to the host computer via a Hewlett-Packard General Purpose Input Output (GPIO) parallel interface.

3.4 Computer System - Hardware

Data collection and analysis is performed using a Hewlett-Packard 9826 computer. Available memory includes 1.3 Mbytes of internal random access memory (RAM), four platters of 1.2 Mbyte hard disk storage, one 1.2 Mbyte eight inch floppy disk drive, and a 256 Kbyte five inch floppy disk drive. A graphics compatible thermal printer and a Decwriter II dot matrix printer are used for hard copy output. An eight pen vector plotter is also available.

All of the peripherals, except for the Decwriter

printer, are interfaced to the computer using HP's General Purpose Interface Bus (GPIB), also known as the IEEE-488 standard. The Decwriter II is connected using an RS-232C interface. The GPIO interface used to control the DAM is also interfaced using the GPIB bus.

3.5 Computer System - Software

HP Pascal Version 3.0 is the operating system used by the computer. Figure 3.3 shows a block diagram of the software system. Two 68000 assembly language routines are used to control the DAM. All of the software is currently written in Pascal, except for the two assembly programs.

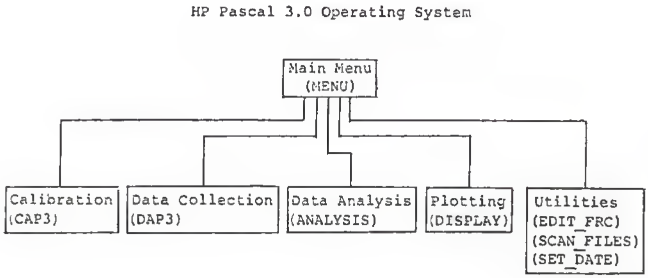


Figure 3.3. Block diagram of the respiratory analysis software.

There are four main programs called by the analysis system: DAP3 (Data Acquisition Program, version 3), CAP3 (Calibration Acquisition Program, version 3), ANALYSIS3,

and DISPLAY3. There are also four small utility programs. The first utility program is the main menu. Its purpose is to call the other programs. Another utility is used to set the date within the computer. The third utility is used to enter the age, weight, and height of all subjects being tested using the system. The last utility is used to search disk directories for respiratory and calibration data files. The operating instructions for all eight programs are included in Appendix A, Operators Guide to the Breath-by-Breath System. Other appendices include the appropriate program documentation.

3.5.1 Data Acquisition Program - DAP3

The data acquisition program is used to collect respiratory data from a subject. A maximum of 90,000 samples can be collected from each of the four channels. If the default sampling rate of fifty Hertz is used then up to thirty minutes of data can be collected. Collected data can be stored on any of the available disk drives, assuming sufficient space is available.

3.5.2 Calibration Acquisition Program - CAP3

The calibration program is used to calibrate all four channels of the DAM. Very little time is required to calibrate the two GMS channels. The thermocouple should be calibrated monthly. The PTM, however, requires a considerable amount of time to calibrate.

3.5.3 Data Analysis Program - ANALYSIS3

The data analysis program is used to analyze previously collected respiratory data. A calibration file must be loaded before a set of respiratory data can be analyzed. The user may choose to analyze the entire set of data or only a portion. If the entire set is analyzed then the display program may be used to do simple statistical analysis or plot the results. If only a portion is analyzed (e.g. window of steady-state data), then the results may be compared to standardized techniques such as the use of a Douglas bag. Real-time analysis is not currently available.

3.5.4 Plotting and Display Program - DISPLAY3

The display program is used to plot or do simple statistical analysis on previously analyzed data. Up to twelve sets of analyzed data can be combined to form composite plots. The composite plots can be used to compare the responses of different subjects or work loads. Several averaging techniques are available for filtering the data. Refer to Appendix A for example output from the display program.

3.6 Problems with the Previous System

Version 2 of the breath-by-breath measurement system contained several cumbersome features. Operators were required to have a thorough understanding of the

respiratory measurement system's hardware and software. Two operating systems were required to run the programs. Users needed to know which operating system was required for each program. In order for the BASIC programs to analyze collected data it was necessary to convert ASCII files created by the Pascal programs into a BASIC compatible format. Converting the file formats was a time consuming task, requiring almost as long as data analysis.

Version 2 of the respiratory analysis system also required a considerable amount of time to analyze data. For example, for nine minutes of data, approximately thirty minutes were required before results were available.

Finally, the response of the PTM was assumed to be linear in all previous versions of this computerized respiratory measurement system. The calculated respiratory variables are dependent on accurately determining inspiratory and expiratory gas volumes. As mentioned in Section 3.2.3, the flow versus differential pressure curve of the pneumotachometer was found to be non-linear at flow rates less than one liter per second. Calibration of the PTM originally was performed using a Harvard fixed volume respirator. The maximum flow rate produced by the respirator is less than one liter per second, which is in the nonlinear portion of the flow calibration curve. The maximum flow rate expected from a

subject exercising at a submaximal level is four liters per second, which is greater than the flows that were previously used for calibration.

IV. IMPLEMENTATION OF PASCAL OPERATING SYSTEM

The first step in implementing the research goals was to change the operating system and programming language from HP BASIC to HP Pascal 3.0. This step was accomplished first because the other goals assume the use of only one operating system.

4.1 Selection of One Operating System

The 200 series of Hewlett-Packard computers has two operating systems. HP's Pascal operating system was chosen over the BASIC operating system for several reasons. Speed of execution and improved file support are the main justifications for using the Pascal environment. There are some disadvantages in eliminating the BASIC code. The tradeoffs between the two implementations are presented here.

4.1.1 Advantages of the Pascal Implementation

Programs written in Pascal are much faster than BASIC programs. The main reason for this speed difference is that Pascal is a compiled language while BASIC is interpreted. Compiled code is inherently faster than interpreted code. Using only one operating system also

eliminates the time consuming task of converting data from Pascal to BASIC formats. The only method HP provides for BASIC and Pascal to share data is via ASCII files. ASCII files are a slow and inefficient means of storing numeric data compared to binary files.

The Pascal implementation of the computerized respiratory measurement system is considerably easier to operate. Menus provide a simple way to select functions. Having only the Pascal operating system eliminates confusion resulting from having to use two different systems. Error trapping has simplified problems associated with disk file support. In addition, programs no longer fail if a disk error occurs.

HP's Pascal system provides excellent file support. Using the filer program, it is relatively easy to make backups of individual files or entire disks. Directory maintenance is simplified by using wild-card characters in file names. Wild-card characters allow users to list or delete any group of files with similar file names. There are also several ways to indicate which disk media to use. Refer to Appendix B, Using the HP Pascal System, for additional information on using the system filer and other features of the Pascal operating system.

4.1.2 Disadvantages of the Pascal Implementation

There are many reasons BASIC was chosen for the

original implementation of the analysis program. HP BASIC provides a good environment for program development. Some of the advantages of BASIC are due to the interactive nature of interpreted languages. The BASIC language supports the computer peripherals better than the Pascal language. Also, Pascal is more susceptible to operator errors than BASIC.

The Pascal system text editor must be used to edit all Pascal programs. Editing files is the only operation that may be performed while using the editor. For example, if the programmer wishes to obtain a directory listing, then the file being edited must be saved, the editor exited, the system filer executed, and then a directory listing requested. BASIC is much simpler because it allows directory listings to be made at any time using the CATALOG command.

Only one printer can be defined in a Pascal system. Redefining the system printer requires editing and recompiling the operating system configuration program, which is not a trivial task. A user may change the default printer at any time in BASIC by executing the PRINTER IS command. The thermal printer is currently defined as Pascal's system printer. This assignment is made so that graphic images can be sent to the printer. The Decwriter II is unable to display graphic images.

Programs that are written using HP Pascal can abort

unexpectantly due to minor user errors. For example, if a numeric value is requested by a program, and the user accidentally presses an alphanumeric key, then the program will abort. The user must then re-execute the program. This problem is circumvented by using a string variable for all user responses. A function which converts the alphanumeric string into a numeric value is then written that ignores all non-numeric characters.

Error trapping is available in Pascal as an aid to handle unexpected run-time errors. A run-time error is any error that occurs while a program is executing. File access is a typical situation where error trapping is useful. If a disk is full or a requested file is not found then an error trapping procedure informs the user that an error occurred and recommends possible corrections. Error trapping is used to protect all of the respiratory analysis programs from disk errors.

The interactive nature of BASIC permits programmers to implement algorithms easily. And, because BASIC is interpreted, fine tuning and debugging of algorithms is simplified. However, after perfecting algorithms it is necessary to improve operating characteristics. Converting all of the code to Pascal greatly improves the performance of the measurement system.

4.2 Simplification of the Operating Procedure

The use of a single operating system simplifies the procedures required to collect and analyze data. However, changing the programming language is not enough to make the system easier to use. The menus previously used to select programs and functions were often cryptic. All of the menus are now less cryptic and contain error checking to ensure valid entries.

A subtle problem with Version 2 of the respiratory measurement system was one of determining if default answers were available. Often, selecting a default answer to a question caused an error to occur later in the program. A typical example of unsatisfactory default values is when the user must enter a file name. Sometimes if the ENTER is pressed when a file name is requested then the program aborts and gives a file name error. Another potential error is when two sections of code use the same variable for storing file names. There is always the possibility that if a default response is used then a valid file might be destroyed.

The convention for naming files is different from the previous version of this measurement system. Now the operator only enters part of a file name. The actual file name used by the system is completed in software by appending prefixes. For example, each channel collected by the data acquisition program is stored in a separate

file. Previously, the user had to enter a file name for each channel of data. Now only one file name must be entered into the program. The file name used to store the carbon dioxide data is preceded with a "C", the file name for the oxygen data is preceded by an "O", the file name for the gas flow data is preceded by an "F", and the file name for the temperature data is preceded by a "T". Analyzed breath-by-breath data can also be stored in disk files. The default file name is the same as the name used for the binary data except an "R" precedes the respiratory data's file name and an "A" precedes the file name used to store the pertinent analysis parameters. Figure 4.1 illustrates the file name prefix technique with an example.

File name entered by the user:	1014G150
Carbon dioxide file name:	C1014G150
Oxygen file name:	O1014G150
Flow file name:	F1014G150
Flow temperature file name:	T1014G150
Analyzed respiratory data file name:	R1014G150
Useful calibration information:	A1014G150

Figure 4.1. Example illustrating the file name prefixes that are generated by the analysis software.

Menu selection of options has also simplified the analysis task. Previously up to sixteen questions had to be answered before data could be analyzed. Normally, identical responses are entered by the operator each time

the analysis program is run. All of the questions are now available by using a separate menu selection. If the operator wishes to change the default settings of these questions he may, otherwise the program assumes the default answers.

Now, the only questions that the user must answer before analyzing a set of data are the starting and ending data points and whether the analyzed results are to be stored on a disk file.

4.3 Increasing the Execution Speed

The majority of improvements in execution speed are the result of using a compiled language instead of the interpreted BASIC. Analysis time is also reduced because the collected data files do not need to be converted from ASCII format to the binary format required by BASIC. Section 5.2 details the improvements in execution speed between Versions 2 and 3 of the respiratory measurement system.

4.4 Improving the Measurement Accuracy

After the analysis algorithms were refined by Masters (3), the calibration procedure was identified as needing improvement. Previously, all four channels of the DAM had to be calibrated prior to each run. Calibration required two operators and twenty minutes to perform. The calibration procedure has been modified. The method of

calibration for the mass spectrometer is unchanged. However, the procedures for calibrating the thermocouple and PTM have changed considerably. For a detailed description of the calibration procedure used for all three devices refer to Appendix A, Operators Guide to the Breath-by-Breath System.

4.4.1 Thermocouple

Thermocouple calibration in Version 2 of the calibration program required three water baths, ranging in temperature from twenty to forty degrees Celsius. One measurement was made at each bath temperature and a second order polynomial was fit to the results. The polynomial coefficients became the calibration factors of the thermocouple. The thermocouple had to be calibrated before each data collection run.

Now, in Version 3, five measurements are taken at four different bath temperatures. The twenty calibration points are then fit to a second order polynomial using a least-squares curve fitting technique. If a few of the calibration points are in error then the temperature calibration is not seriously inaccurate. The thermocouple only needs to be calibrated once each month, assuming the DAM is not altered.

4.4.2 Flow

The PTM was previously calibrated using a Harvard

respirator pump. During calibration, the pump was set to its maximum speed and flow calibration factors were determined for both inspiration and expiration.

A flow calibration factor is calculated by dividing the actual pump volume by the integral of the difference between the binary differential pressure and binary zero flow. A calibration factor is calculated for both inspiratory and expiratory pump flows. If the flow transducer is linear then the same integral should be calculated regardless of the flow rate used for the calibration.

In actual use, different flow calibration factors are obtained for different flow calibration rates, indicating that the transducer is not linear. Figure 4.2 shows a plot of the calibration factor versus calibration flow rate for both inspiratory and expiratory flows. The calibration factors plotted in Figure 4.2 are determined by forcing a known volume of saturated room air through the PTM at different flow rates. It is obvious from Figure 4.2 that the flow calibration factors are a function of the flow rates used for calibration.

Version three of the calibration program uses a large syringe for calibrating the PTM. The syringe is set to a known volume and pumped manually. A large syringe is used so that the actual volume displacement of the syringe is closer to an actual tidal volume. The syringe also

enables the PTM to be calibrated over the entire expected operating range.

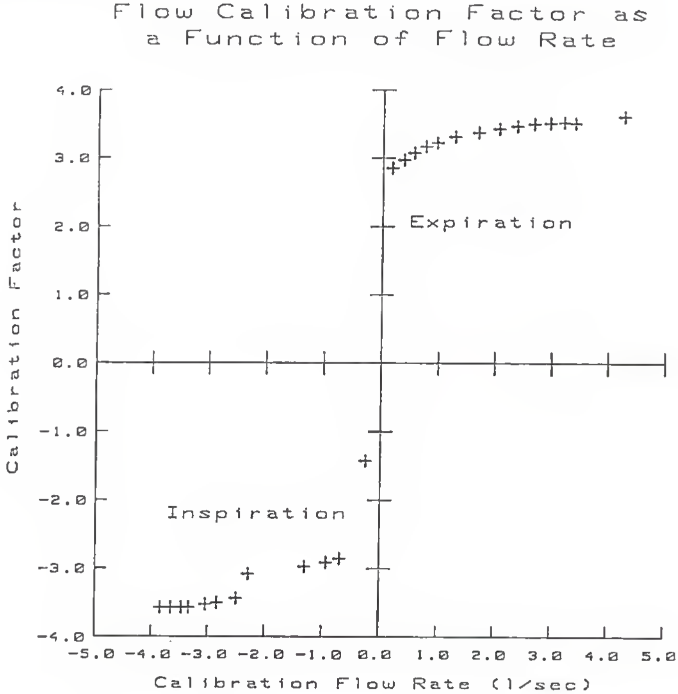


Figure 4.2. Flow calibration factor plotted as a function of calibration flow rate.

Eight flow calibration points are taken for both inspiratory and expiratory flows. A third-order polynomial is least-squares fit to the eight data points to give the calibration factor as a function of the average differential binary flow.

4.5 Produce Portable Code

If the computerized respiratory measurement system is ever adapted to another research project then the requirement of a specific computer system may hinder the adaptation. Although the Hewlett-Packard 200 series computers are well suited to this project, they are rather expensive. Future adaptations of the computerized respiratory measurement system may require the use of an IBM PC or compatible as a controller. Although these machines are not as fast as the Hewlett-Packard 200 series, a large volume of third party software, hardware, and technical support is available for the MS-DOS based computers.

The main problem encountered when considering a change to a different computer concerns the availability and compatibility of a Pascal compiler for the new computer. The compiler must be able to compile programs that were written for a different compiler on a different computer. Pascal compilers are available for most computers, but the versions differ in the way they handle non-standard devices (e.g., a DAM or voltmeter) and control graphics devices. Most Pascal programs that do not contain graphics, device dependent input and output (I/O) instructions, or other language extensions should run without modifications on any computer system that has a Pascal compiler.

If the data collection and analysis programs are transferred to another computer system then all of the procedures that control non-standard devices or graphics will need to be modified. All of the programs in Version 3 of the analysis programs have been written with the non-standard I/O instructions and the graphics statements in separate procedures. Because the non-standard procedures are separated, the editing process will be simpler if the analysis programs are ever transported to another computer.

Even though there are problems associated with the differences among different Pascal compilers, different BASIC interpreters are more varied. Extensive modifications to the BASIC language are typically necessary because standard BASIC is very limited in the size and types of variables that can be defined. HP, as well as other computer manufacturers, has added many non-standard extensions to the BASIC language in order to increase the power of their implementations. Because the different BASIC interpreters have many non-standard extensions, transporting a BASIC version of the analysis programs would be more difficult than transporting Version 3 of this Pascal implementation.

4.5.1 Turbo Pascal

Turbo Pascal, marketed by Borland International,

Inc., is a Pascal compiler available for MS-DOS based personal computers (PC) such as the IBM PC and Zenith Z-150. Turbo Pascal is currently the de facto Pascal standard and is very similar to HP Pascal. It is the de facto standard because it is inexpensive and contains sufficient language enhancements to fully utilize the resources of a PC.

Unfortunately, Turbo Pascal will not adequately support the respiratory analysis software. The Turbo Pascal editor only accepts source code smaller than 64 Kbytes. Also, the maximum compiled code size is also limited to less than 64 Kbytes. These limitations could be circumvented by using program overlays and chaining, but the necessary code revision would be difficult. If the software is ported to an MS-DOS based computer then Pascal compilers other than Turbo Pascal should be considered.

4.6 Longer Data Runs

Two changes are made to the computerized respiratory measurement system that increase the maximum data collection time from ten minutes to thirty minutes. More RAM has been installed in the computer and the storage technique has been modified so that only half as much memory is required to store each sample.

4.6.1 Increasing RAM

The amount of installed RAM has been increased to 1.3 Mbytes. Programs written in Pascal can use the additional memory by allocating dynamic arrays. Dynamic arrays are accessed similarly to conventional arrays except pointers are used in place of array subscripts. One should consult any introductory Pascal text for additional information regarding dynamic memory allocation and pointers (8,9).

4.6.2 Changing Variable Type

The DAM returns a 12-bit analog-to-digital conversion each time an input channel is sampled. Version 2 of the data collection and analysis programs stored the data returned from the DAM in integer arrays. Normally, HP Pascal allocates 32-bits of memory for each integer variable. Therefore, when a 12-bit conversion is stored in an integer variable, 20 of the 32 bits (62.5%) are wasted.

The Pascal programming language permits variable type declarations that are unique to specific applications. To reduce the amount of wasted memory, Version 3 defines a subrange of integers that requires only 16-bits of storage. All variables defined using the subrange execute faster and use half as much memory as integer variables (10). When the arrays used to store the DAM data are defined using 16-bit integers, only four bits (25%) are wasted.

Defining the subrange to be 12-bits, the size of the converted data, would not change the storage capacity but would increase the execution time. The storage capacity would not be increased by using the 12-bit integers because Pascal allocates memory in multiples of 16-bits. Execution time would be slower because range checking would be performed each time any of the 12-bit values were accessed.

V. VERIFICATION

System verification requires that the goals of Chapter II be achieved. One of the goals, simplifying the operating procedure, is subjective and will only be appreciated by someone who has used Version 2 of the computerized respiratory measurement system.

5.1 Operating System and Language Conversion.

Due to the quantity of program code, many possibilities for errors arise during the translation from BASIC to Pascal. Verification requires that the same results are obtained regardless of the programming language. Table 5.1 compares the results between Versions 2 and 3 of the analysis software. Some differences in the results are expected and are caused by round-off errors.

Nine minutes (27,000 samples) of data were analyzed to generate the results of Table 5.1. Some of the variables listed in Table 5.1, such as the respiratory quotient, should not be considered physiologically valid because the results are an average of resting, exercise, and recovery data.

Parameter	Units	Version 3	Version 2	Error %
Insp. minute vol.	l/min BTPS	-54.108	-54.095	0.024
Expr. minute vol.	l/min BTPS	52.111	51.956	0.298
O_2 consumed	l/min STPD	-2.028	-2.046	0.880
CO_2 produced	l/min STPD	1.752	1.746	0.344
Alveolar O_2 cons.	l/min STPD	-1.791	-1.789	0.112
Alveolar CO_2 prod.	l/min STPD	1.860	1.860	0.000
Respiratory quotient		0.864	0.853	1.290
Alveolar Respiratory quotient		1.039	1.040	0.096
Insp. tidal volume	l BTPS	-1.998	-1.997	0.050
Expr. tidal volume	l BTPS	1.924	1.918	0.313
Respiratory freq.	breaths/min	27.1	27.1	0.000
Mean O_2 insp.	l/breath STPD	-0.330	-0.330	0.000
Mean O_2 expr.	l/breath STPD	0.255	0.254	0.394
Mean CO_2 insp.	l/breath STPD	-0.002	-0.002	0.000
Mean CO_2 prod.	l/breath STPD	0.067	0.066	1.52
Mean O_2 cons.	l/breath STPD	-0.075	-0.076	1.32
Mean CO_2 prod.	l/breath STPD	0.065	0.064	1.56

Table 5.1. Comparison between the results of the current and previous implementations for 27,000 data points.

Parameter	Units	Version 3	Version 2	Error %
Insp. minute vol.	l/min BTPS	-76.460	-76.451	0.012
Expr. minute vol.	l/min BTPS	72.731	72.514	0.299
O_2 consumed	l/min STPD	-2.925	-2.952	0.915
CO_2 produced	l/min STPD	2.416	2.409	0.291
Alveolar O_2 cons.	l/min STPD	-2.446	-2.447	0.041
Alveolar CO_2 prod.	l/min STPD	2.598	2.599	0.038
Respiratory quotient		0.826	0.816	1.225
Alveolar Respiratory quotient		1.062	1.062	0.000
Insp. tidal volume	l BTPS	-2.172	-2.172	0.000
Expr. tidal volume	l BTPS	2.066	2.060	0.291
Respiratory freq.	breaths/min	35.2	35.2	0.000
Mean O_2 insp.	l/breath STPD	-0.359	-0.356	0.843
Mean O_2 expr.	l/breath STPD	0.276	0.275	0.364
Mean CO_2 insp.	l/breath STPD	-0.002	-0.002	0.000
Mean CO_2 prod.	l/breath STPD	0.071	0.071	0.000
Mean O_2 cons.	l/breath STPD	-0.083	0.084	1.19
Mean CO_2 prod.	l/breath STPD	0.069	0.068	1.47

Table 5.2. Comparison between the results of the current and previous implementations for one minute of 150 watt steady-state exercise.

Table 5.2 makes a similar comparison using one minute of steady-state exercise data. All of the results listed in Table 5.2 are valid.

It should be recognized that the percent error for the mean oxygen inspired, oxygen expired, oxygen consumed and carbon dioxide produced is calculated using only two significant digits. No significant calculation errors exist between the two implementations.

5.2 Increasing Execution Speed

The time required to analyze a set of respiratory data using Version 2 of the analysis system was considerable. More time was required for data analysis than data collection. Table 5.3 summarizes a typical set of execution times for the two versions. The data presented in Table 5.3 came from analyzing nine minutes of respiratory data sampled at a rate of fifty Hertz. Four times are listed in Table 5.3. The next four sections describe the significance of each time.

	Version 3 (min:sec)	Version 2 (min:sec)
Typical calibration time.	30:00	20:00
Maximum data collection time.	30:00	10:00
Data format change from ASCII to binary for nine minutes of data.	0:00	15:00
Typical analysis time for nine minutes of data.	5:45	11:45

Table 5.3. Typical times required for calibration, collection, and analysis of respiratory data.

5.2.1 Calibration

The amount of time needed to calibrate the respiratory measurement system increased by ten minutes. The increase occurs because additional flow calibration points are taken. Section 4.4.2 discusses the changes in the calibration procedure in greater detail. Analysis accuracy is dependent on the quality of the calibration. The only way to decrease the flow calibration time, yet maintain accuracy, will require implementing a partially automated flow calibration system. This option is discussed in Section 6.7.

5.2.2 Maximum Data Collection Time

The maximum amount of data that can be collected is 90,000 samples per channel, which is equal to thirty minutes of data collected at a fifty Hertz sampling rate. The previous maximum was 30,000 samples per channel. The modifications that enabled the increase in the maximum number of samples are described in Section 4.6.

The data collection program is not the only limiting factor for the maximum number of samples that can be collected. The program ANALYSIS3, which is used to analyze the respiratory data, must also be able to load the entire set of data into memory. If more than thirty minutes of data collection are required then either the sampling rate should be decreased or the ANALYSIS3 program

should be modified to load the respiratory data from the disk as it is required.

5.2.3 Format Change from ASCII to Binary

Version 2 of the respiratory analysis software used both the HP BASIC and HP Pascal operating systems. Only one method exists for HP's BASIC and Pascal operating systems to share data files. Data used by both systems must be stored in ASCII files. A BASIC program (DAPCRUNCH) was used to convert the ASCII files created by the Pascal data collection program (DAP2) into BASIC's binary integer representation. Because the BASIC operating system is not used in Version 3, this conversion time is eliminated.

5.2.4 Typical Analysis Time

This value represents the amount of time required to analyze four channels of 27,000 data points. Corrections made to the data include conversions between STPD and BTPS conditions, adjustments for temperature and viscosity changes, and FRC corrections.

Note from Table 5.3 that the analysis time for Version 3 is less than the data collection time. This indicates that real-time analysis may be possible. Adapting to real-time analysis is discussed in detail in Section 6.3.

VI. FUTURE MODIFICATIONS

Several additional improvements are suggested for the computerized respiratory measurement system. Some of the following recommendations, such as spooling and real-time analysis, will significantly change the measurement system. Other modifications, such as using a turbine for flow measurements, may improve the measurement accuracy.

6.1 Spooling Data to Disk

Theoretically it is possible to store data in a disk file as it is collected. Recording the data directly to the disk instead of storing it in memory would increase the maximum number of samples that can be taken to the capacity of the destination disk. Figure 6.1 is a block diagram of a system used to test spooling of data directly to disk. A sine wave of approximately one half Hertz is applied to an input channel of the DAM. Samples are taken and then immediately stored on the hard disk. The hard disk is used for this test because it has the fastest response time of all available drives. Figures 6.2 and 6.3 show data recorded using sampling rates of ten and fifty Hertz.

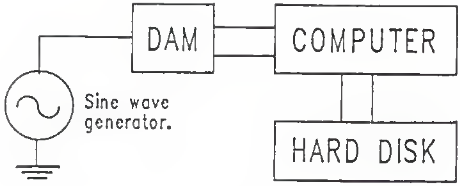


Figure 6.1. Block diagram of the system used to test spooling data directly to disk.

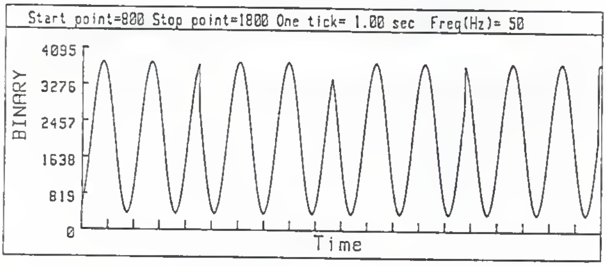


Figure 6.2. Sampled data collected by spooling directly to disk at a sample rate of 50 Hz.

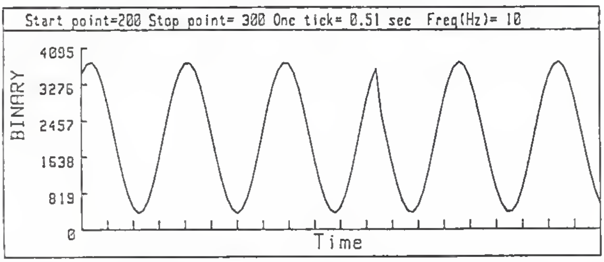


Figure 6.3. Sampled data collected by spooling directly to disk at a sample rate of 10 Hz.

Notice that there are missing samples in both signals. The missing samples are due to the polling

technique that is used for data acquisition. Section 6.1.1 describes the polling technique and why the samples are lost. An interrupt controlled method of data collection is described and proposed as an enhancement in Section 6.2.

6.1.1 Polling

The DAM is controlled by a software polling technique. Polling is a method where a program waits for a timer to indicate that samples are to be taken. Figure 6.4 represents a flow diagram used for polling at a set rate.

The reason samples are lost when spooling the data directly to a disk file is simple. When a program sends information to a disk file, the operating system temporarily stores the data in an output buffer. When the buffer is full, the operating system requests service from the drive controller and then passes the data to the disk drive. Sampling periods are missed while the output buffer is transferred to the disk because a significant amount of time is required to send the buffer's data to the drive. The lost sample problem associated with spooling can be solved by using an interrupt driven data acquisition module, which is described in the next section.

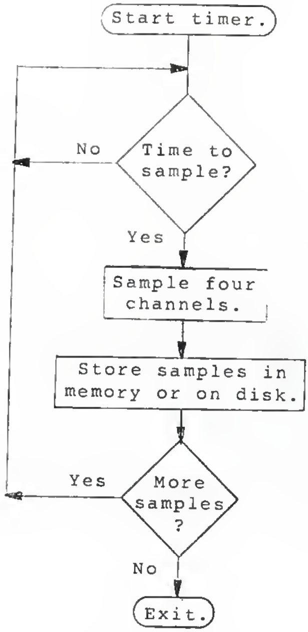


Figure 6.4. Flow diagram of the polling technique of data collection.

6.2 Interrupt Controlled DAM

An interrupt controlled DAM would simplify future enhancements of the computerized respiratory measurement system. Interrupt control would enable the system to collect larger sets of data by using a spooling technique of data storage. More data could result from the addition of more channels, increasing the sampling frequency, or

increasing the length of runs.

Interrupt control of the DAM would also facilitate adapting the system to real-time analysis. Real-time analysis would be extremely difficult relying on the polled DAM currently used.

An interrupt controlled method of data collection is more complex than polling but it solves the problem of missing samples. When it is time to sample, the DAM's hardware interrupts the controlling computer. The operating system acknowledges the interrupt request and executes an interrupt service routine. The interrupt service routine controls the conversion of the DAM channels and stores the results in a buffer. After completing the conversions, control of the computer is returned to the program that was running prior to the interrupt request. The Pascal operating system allows the programmer to write interrupt service routines in either Pascal or 68000 assembly language. Unfortunately, the current controlling hardware of the DAM is not capable of interrupting the HP 9826 computer.

The HP 9826 computer contains a timer that can be configured to interrupt the computer at a rate of 100 Hertz. This timer could be used to provide interrupts with a period of multiples of 10 msec. Refer to the System Devices section of the Procedures Guide (11) for example programs that use this technique.

6.3 Real-Time Analysis

The Pascal language, as implemented on the Hewlett-Packard computer system, is fast enough to provide real-time analysis of the respiratory variables. Instantaneous readings would be useful in determining when an exercising subject reaches steady state or exceeds potentially dangerous levels of exercise. Graphical output of carbon-dioxide production and oxygen consumption could be used as an indicator of steady-state. Adapting to real-time operation is the next logical step in the evolution of this computerized respiratory measurement system.

Table 6.1 lists the amount of computer time required to collect four channels of data at sampling rates of 350, 50, and 30 Hz. The maximum sampling rate of the DAM is 350 Hz. Fifty Hz is listed in the table because that is the sampling rate currently used. Thirty Hz is included as a possible lower sampling frequency. In order for real-time analysis to be possible, the computer must be able to process incoming data faster than they are collected.

6.3.1 Problems Associated with Real-Time

Currently, the computer collects data from the DAM using a polling technique. An interrupt scheme is preferred for real-time analysis. Sections 6.1.1 describes the polling technique while Section 6.2

describes the interrupt technique. Because the DAM is polled, many problems must be overcome before real-time analysis becomes a reality.

	Sampling Rate (Hz)		
	350	50	30
Sampling period (msec)	2.86	20.0	33.3
time controlling DAM (%)	100.0	14.3	8.6
time waiting (%)	0.0	85.7	91.4
For nine minute run:			
Sampling time (min:sec)	9:00	1:17	0:46
Waiting time (min:sec)	0:00	7:43	8:14
Analysis time (min:sec)	40:30*	5:45	3:26

* Approximated. System is not capable of taking this many data points.

Table 6.1. Computer time required for data collection and analysis.

The main problem with adapting to real-time analysis involves calculations that are only made at the end of each breath. Numerical integration is used to calculate the inspiratory and expiratory gas volumes. During analysis, each data point is checked to see if it is the end of the current inspiratory or expiratory cycle. If the point is not a separator between inspiration and expiration then its incremental volume is added to the total for that breath and the program proceeds to the next sampling point. If the point is the end of inspiration then half of its volume is added to the inspiratory cycle and half to the upcoming expiratory cycle.

When the point is the end of expiration, half of its

volume is added to the expiratory cycle and half to the next inspiratory cycle. This point also indicates the end of the current breath and the beginning of the next breath. Before the algorithm continues to the next sampling point, the respiratory variables that are dependent on volumes, such as the respiratory quotient, are calculated. If the time delay of the GMS is to be calculated for each breath then the new time delay must also be determined before the end of the current sampling period.

If the analysis is running in real-time then results must also be plotted on the display after each breath. Many time consuming floating point calculations are required to plot a single point on a graphical display. The time required to plot a point on a graphical display may be the primary obstacle in a real-time system using a polled DAM.

Clearly, more time is required to analyze the results between breaths than during the breath. Because the data acquisition system is polled, all of the end-of-breath calculations must be made within one sampling period. If the calculations are not completed by the end of the current sampling period then the next data point will be missed.

6.3.2 Possible Solutions to Real-Time Problems

The breath-by-breath calculation of the GMS time delay could be replaced with a fixed time delay that is calculated prior to the run. The algorithm that is currently used to determine the breath-by-breath time delay could be used to determine the best fixed time delay. Analysis of the data after the run would still use the breath-by-breath method.

A slower sampling rate could be used. The slower rate would provide more time for calculations between samples. Using a slower sampling rate may reduce the measurement accuracy and should be investigated further. Section 6.4 proposes a study to determine the optimal sampling rate.

Plotting the results of a breath could wait until after the next breath starts. After calculating all of the respiratory variables, a flag is set within the program. After the next breath starts and the end-of-breath calculations are completed, the program could plot one of the variables. Other variables that require plotting could be processed during subsequent intervals. This technique is not simple but it is a possible solution to real-time analysis using the polled DAM.

Replacing the polled DAM with an interrupt driven DAM would minimize the problems associated with the end-of-breath calculations. If the computer is making a

calculation when an interrupt occurs then the DAM will be serviced and the new data points placed in buffers for later analysis.

Because the computer has a net surplus of computing time, it could empty a backlog of respiratory data points prior to the start of the next breath. Breath-by-breath GMS time delay calculations could also be maintained because the algorithm permits the extra calculation time required for the end-of-breath data points.

6.4 Optimal Sampling Rate

The respiratory measurement system currently samples the four analog signals at a rate of fifty Hertz. However, Bernard (12) claims that a sampling rate of thirty Hertz is sufficient in a simulated environment. If a thirty Hertz sampling rate is used, nearly twice as much data could be collected at one time than is possible at fifty Hertz. Analysis time would also decrease substantially with a slower sampling rate. A lower sampling rate could also facilitate a real-time analysis system because the time available between samples for calculations and plotting would increase.

Increasing the sampling frequency above fifty Hertz may also provide the optimal rate. A faster sampling rate would increase the total amount of available information, possibly increasing the measurement accuracy. Accuracy

would increase because a better GMS time delay could be determined. The GMS time delay is an important variable in determining the volumes of carbon-dioxide, oxygen, and nitrogen that are inspired and expired by the subject. More data points would also provide a better representation of the respiratory signal, permitting a more precise numerical integration. An integration that is more exact should yield an improvement in the calculated results.

6.5 Turbine for Flow Measurement

Replacing the PTM with a positive displacement flow device, such as a turbine, may increase the accuracy of gas volume measurements. The turbine, unlike the PTM, is supposedly not affected by changes in gas temperature or viscosity (13).

Turbines use a rotating blade for measuring volumes. Figure 6.5 represents a simplified diagram of a turbine. Gas flow causes the turbine blades to spin. Infra-red LED (light emitting diodes) transmitters and receivers are used to determine the number of blade rotations and the direction of rotation. As the blade rotates it sequentially blocks the infra-red light signals. The resulting digital signal is then sent to a counter for direct reading by a computer. The digital signal could also be processed in hardware to provide an analog representation of either volume or flow.

Rotational inertia of the turbine blade may be a problem. The inertia effects must be studied to determine if they add a significant amount of error.

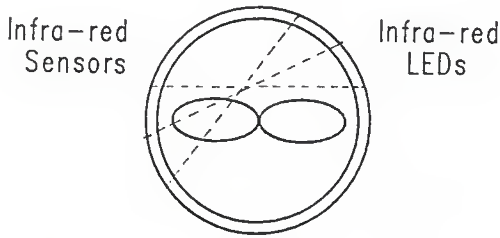


Figure 6.5. Schematic diagram of a turbine flow transducer.

6.6 Heart Rate Measurements

Adding a heart rate channel to the DAM would simplify the procedure used for heart rate measurement. Currently, the heart rate is obtained using a telemetry system with the transmitter worn by the subject. A receiver stores the average heart rate every thirty seconds. After the exercise session, an operator must copy the heart rate values into a notebook.

The current receiver could be redesigned so that the incoming heart beat pulses are sent directly to the DAM. There should be sufficient processing time available for determining the heart rate in real-time. Displaying the heart rate on the computer's CRT during data collection would permit easy monitoring of the subject. A software

alarm could also be developed that would signal the operator if the heart rate exceeded a preset threshold.

6.7 Flow Calibration Improvements

Calibrating the PTM requires a significant amount of operator time. A large syringe is currently used because it is able to produce flow rates that include the entire operating range of the PTM. Eight calibration points are taken with the syringe, with each point representing a different average flow rate. The manually operated syringe is not an efficient method of calibration and should be replaced. One possible replacement is a large, computer-controlled, oscillatory pump. A constant flow calibration method using a tank of compressed air may also provide accurate results.

A computer-controlled oscillating pump would greatly simplify the calibration procedure. The average flow rate of the pump would be changed by varying the rate of the pump. At least eight different pump rates are required for accurate flow calibration.

VII. CONCLUSIONS

Optimization of the computerized breath-by-breath respiratory measurement system provides the following conclusions. Most of the conclusions are possible due to the exclusive use of the HP Pascal operating system and programming language.

1. The HP Pascal operating system, which incorporates the Pascal programming language, is suitable for the collection and analysis of breath-by-breath respiratory data. Although the Pascal operating system is more difficult to use than the BASIC system, the advantages permit easier operation of the respiratory measurement system and faster execution times.

2. The operating procedure of the system is improved by adding default values to all questions. Menu selection of options permits easier learning of the system. Because only the Pascal system is used, it is not necessary for the operator to be familiar with the BASIC environment. Having only one operating system also eliminates converting disk files into a format

that is readable by BASIC.

3. Execution times can be decreased considerably by using a compiled language such as Pascal. Although more difficult than BASIC for program development, the compiled Pascal programs are much faster than the interpreted BASIC programs.

4. Measurement accuracy can be increased by modifying the calibration technique. The thermocouple only needs to be calibrated monthly. Because the flow transducer is not linear, eight calibration points are taken at different average flow rates.

5. The Pascal source code is portable to other computer systems. The majority of the software is written using standard Pascal. Only the routines that use graphics or system peripherals will require modifications if the programs are ported to a different computer. The Pascal compiler used on a different computer must be able to handle large source programs. Turbo Pascal, which runs on IBM PC's and compatibles, would not be able to compile the respiratory analysis programs.

6. The maximum amount of data that can be collected at one time can be increased. The increase is facilitated by changing to a data type that uses only 16-

bits of storage and by increasing the amount of available memory.

7. Further enhancement of the respiratory measurement system by adding real-time analysis is possible. The modifications may require redesigning the data acquisition module to support interrupt data collection techniques.

VIII. BIBLIOGRAPHY

(1) E. E. Creel, Measurement of Breath-by-Breath Oxygen Consumption and Carbon Dioxide Production in Exercising Calves, Master's Thesis, Kansas State University, 1982.

(2) L. E. Riblett, Jr., A Computer-Based Instrumentation System for Measurement of Breath-by-Breath Oxygen Consumption and Carbon Dioxide Production in Exercising Humans, Master's Thesis, Kansas State University, 1984.

(3) M. H. Masters, A Computer-Based Respiratory Measurement System and a Temperature Transducer for Monitoring Respiratory Flow Temperature, Master's Thesis, Kansas State University, 1985.

(4) P. Astrand, Work Tests with the Bicycle Ergometer, Varberg, Sweden, Monark-Crescent AB.

(5) R. L. Pieschl, "The Application of FRC Corrections to Respiratory Gas Volumes Calculated Using Breath-by-Breath Techniques", Personal Communication, 1987.

(6) Equipment specifications, Hans Roudolph, Inc., Kansas City, MO, Rudolph Pneumotachometer, Model number 3813, Serial number 38-1432, 1984.

(7) L. W. Gateno, Design of a Versatile, Multi-Channeled, Data Acquisition Module, Master's Thesis, Kansas State University, 1980.

(8) G. M. Schneider, S. W. Weingart, and D. M. Perlman, An Introduction to Programming and Problem Solving with Pascal, second edition, John Wiley and sons, 1982.

(9) Borland International, Inc., Turbo Tutor, A Turbo Pascal Tutorial, second edition, Borland International, Inc., 1984.

(10) The Pascal Handbook for the Series 200 Computers, third edition, Fort Collins, Hewlett-Packard Desktop Computer Division, Sybex, 1982.

(11) Pascal 3.0 Procedure Library for the HP 9000 Series 200 Computers, System Devices, first edition with update, Fort Collins, Hewlett-Packard Desktop Computer Division, 1984.

(12) T. E. Bernard, "Aspects of On-line Digital Integration of Pulmonary Gas Transfer", J. Appl. Physiol., vol. 43(2), pp 375-378, 1977.

(13) N. Lamarra, Ventilatory Control, Cardiac Output, and Gas-Exchange Dynamics During Exercise Transients in Man, Dissertation, University of California Los Angeles, 1982.

ACKNOWLEDGEMENTS

I would like to thank the Department of Electrical and Computer Engineering for financial assistance during my research.

I would also like to thank my committee members Dr. R. A. Dyer and Dr. M. R. Fedde for their recommendations and assistance. Special thanks goes to my major professor Dr. R. R. Gallagher. Without his support this thesis would not have been possible.

It was my parents and family who encouraged me to continue my education and gave me much needed emotional support. I dedicate this thesis to my family because they will always be special to me.

APPENDICES

TABLE OF CONTENTS - APPENDIX A

OPERATORS GUIDE TO THE BREATH-BY-BREATH SYSTEM

LIST OF FIGURES - APPENDIX A

LIST OF TABLES - APPENDIX A

APPENDIX A

OPERATORS GUIDE TO THE BREATH-BY-BREATH SYSTEM

This appendix contains a guide to the operation of the computerized breath-by-breath respiratory analysis system. Only the operation of the analysis software is presented in this appendix. Appendix B contains a brief description of the operating system.

A.1 General Information

A.1.1 Intended Audience

This users guide is written with the assumption that the user has had no experience using the computerized breath-by-breath respiratory measurement system. The programs described in this appendix have been sufficiently debugged so they can not cause any damage to the computer or instrumentation system. If the computer should unexpectedly "crash" then the computer's power should be shut off for a few seconds then turned back on. Working with a partner will make learning the system easier.

A.1.2 Answering Yes/No Questions

Many questions asked by the analysis software require a simple yes/no answer. To respond "yes" to a question, press the "Y" key then the ENTER key. For a "no" response, press the "N" key then the ENTER

key. Either upper or lower case characters may be used.

A.1.3 Default Answers

Default answers are displayed in square brackets and may be selected by simply pressing the ENTER key. For example, if a question states

DO YOU WANT TO EAT LUNCH? ([Y]/N): __

then the default answer is yes because the "Y" is in brackets.

A.1.4 Selecting Mass Storage Devices

There are several disk drives that can be used for data storage. The prompt in Figure A.1 is displayed whenever the user must select a mass storage device.

```
You may choose one of the mass storage
locations below.

    1.   The hard disk, Platter 2.
    2.   The hard disk, Platter 3.
    3.   The 8 inch floppy disk.
    4.   The 5 inch floppy disk.
    5.   The RAM disk.

Which device do you want to use? [3]: __
```

Figure A.1. Mass storage device list.

The default response is selection number three, the eight inch drive. The five inch drive and the RAM drive have limited storage and should only be used if the operator is familiar with HP Pascal.

A.1.5 Power-up Sequence

The computer must be the last device turned on. If the computer

is turned on before the two external drives, only the internal five inch drive can be used. The Perkin Elmer Gas Mass Spectrometer (GMS) should remain in the standby mode when not in use. If power is removed from the GMS then up to eight hours are required to obtain an operating vacuum. The Godart pressure transducer should also be left on continuously as it requires at least twenty-four hours to warm up. The black, zero-suppression box that is used by the GMS requires a long warm up period and should be left on at all times.

The following eight steps outline the power-up sequence.

1. If only data analysis is to be performed then proceed to step number 5.

2. Turn on the Godart pressure transducer, GMS zero suppression box, and the Data Acquisition Module (DAM). The power switch for the DAM is located on the left edge of the Tectronics power supply which is below the DAM. Allow 24 hours for all three devices to warm up.

3. Turn on the GMS (from STANDBY).

4. Turn on the pneumotachometer (PTM) heater. The heater is the brown box mounted at the top of the gray rack. Allow fifteen minutes to warm up.

5. Turn on the hard drive and 8 inch floppy drive.

6. Place the 5 inch disk marked "Pascal 3.0 System Boot / Breath-by-Breath" in the computer's 5 inch floppy drive.

7. Turn on the computer. The Pascal operating system will boot, requiring about two minutes.

8. When asked for the current date, enter the date using the format MM/DD/YY. For example: If the date is January 5, 1987 then enter 01/05/87. Pressing ENTER without entering a date leaves the system date unchanged. Only valid dates will be accepted by the program.

   The main menu for the breath-by-breath system will then be displayed. Figure A.2 shows the main menu.

A.2 Main Menu

The main menu is used to select from the different functions of the program, such as system calibration or data analysis. A brief description of the seven choices follows. Detailed instructions for each selection follow later in this appendix. Figure A.2 shows the main menu as it appears on the computer's display.

```
*******************************************
*                                         *
*                MAIN MENU                *
*                                         *
*     Breath-by-Breath Analysis System    *
*                                         *
*                                         *
*  1) Calibration      - HARD1:CAP3       *
*  2) Data Collection  - HARD1:DAP3       *
*  3) Data Analysis    - HARD1:ANALYSIS   *
*  4) Display/Plotting - HARD1:DISPLAY    *
*                                         *
*  5) Subject Info.    - HARD1:EDIT_FRC   *
*  6) Set the DATE     - HARD0:SET_DATE   *
*  7) Search for files - HARD1:DISK_SCAN  *
*                                         *
*  8) Exit to SYSTEM                      *
*                                         *
*                                         *
*  Enter your choice: __                  *
*                                         *
* Version:  2-Apr-87                      *
*******************************************
```

Figure A.2. Main menu for the breath-by-breath system.

1. Calibration This selection is used to calibrate the breath-by-breath system. Refer to Section A.3 for detailed instructions on system calibration.

2. Data Collection This selection is used to collect data from an exercising subject. Collected data is stored in disk files for later analysis. Refer to Section A.4 for additional instructions

on data collection.

3. Data analysis This selection is used to analyze raw data obtained by the data collection program. Refer to Section A.5 for additional instructions on data analysis.

4. Display/Plotting This program is used to manipulate previously analyzed sets of data. Averaged results may be printed in tabular form or plotted on either the CRT or plotter. Refer to Section A.6 for additional instructions on the plotting features.

5. Edit Subject Information is used to maintain a list of default age, weight, and height values for each subject. The three parameters are used to estimate the functional residual capacity (FRC) of the subject. Refer to Section A.7 for additional instructions on the edit subject information program.

6. Set the Date This selection is used to change the date stored in the computer. The current date is recorded by the operating system each time a file is accessed. Refer to Section A.8 for additional instructions on setting the system date.

7. Search for Files This selection is used to scan a disk for possible data files. The user may specify either calibration files, binary DAM data files, or analyzed respiratory data files. Refer to Section A.9 for additional instructions on searching for data files.

8. Exit to the Operating System This selection returns control of the computer to the HP Pascal operating system. Refer to Section A.10 for additional information about exiting to the operating system and restarting the menu program.

A.3 Calibrating the System

This section describes the calibration procedure developed for the breath-by-breath respiratory measurement system. Calibration is necessary so the analysis program can interpret the gas concentration, flow, and temperature signals. To execute the calibration menu, choose selection one from the main menu. Figure A.3 shows the calibration menu as it appears when first run.

```
********************************************
*                                          *
*         Calibration Program Menu         *
*                                          *
* Calibrate:                  Calibrated   *
*   1) GMS O2 and CO2           No         *
*   2) Thermocouple             Default    *
*   3) Flow                     No         *
*                                          *
*   4) Store Calibration.                  *
*                                          *
*   5) Return to the MAIN MENU.            *
*                                          *
*                                          *
*   Enter your choice: __                  *
*                                          *
*                                          *
* Version:  14-APR-87                      *
********************************************
```

Figure A.3. Calibration menu.

A.3.1 General Information

The right most column of the calibration menu indicates whether the devices listed in the left column are calibrated. The flow transducer and GMS should be calibrated each day. The thermocouple needs to be calibrated once each month. If the thermocouple is using the default calibration then the calibration menu will show "Default".

If the thermocouple is recalibrated then the menu shows "Yes". After the flow and GMS are calibrated, the calibration menu displays "Yes" for the respective device. Both the GMS and flow transducers must be calibrated before the calibration file may be stored on the disk. The following sections describe setting the sampling frequency and step-by-step instructions for each of the calibration procedures.

A.3.1.1 Storage Requirements

Each calibration file requires 512 bytes of disk storage. Because this is such a small amount of storage, several hundred calibration files can be stored on a single eight inch disk. Five inch disks have a default maximum of 80 directory entries. The maximum number of directory entries for a disk can be increased by using the Zap function of the system filer.

A.3.1.2 Setting the Sampling Frequency

Normally a sampling rate of 50 Hz is used for calibration. The user only needs to enter the sampling frequency one time. The program will prompt the user for the frequency the first time that it is required. The prompt requesting the sampling frequency is:

```
ENTER THE SAMPLING FREQUENCY (HZ) [50]: __
```

The default response is 50 Hz. Sampling rates greater than 300 Hz are not recommended. Negative sampling rates are not permitted.

A.3.2 Calibrating the GMS

This menu selection is used to calibrate the CO_2 and O_2 gas

concentration channels. Two gases of known CO_2, O_2, and N_2 concentrations are required. Normally, room air provides one reference and a bottle of 7% CO_2, 13% O_2, balance N_2 provides the other. The calibration for room air is done first.

A.3.2.1 <u>Room Air - GMS Calibration</u>

The following prompts will appear and the user will need to supply an appropriate answer. This guide indicates all prompts from the computer in upper case characters and a typical user responses is underlined. Default responses are displayed in square brackets. Be sure the GMS is turned on.

CONNECT THE GMS PROBE TO ROOM AIR.
ENTER THE PERCENT CONCENTRATION OF CO2 [0.05]: <u>0.04</u>

Place the end of the GMS sampling capillary in an open area of the laboratory. Enter the CO_2 concentration. The CO_2 concentration is found on the right most display of the GMS. The reading should be about 0.05%. Do not enter the percent sign with the value.

ENTER THE PERCENT CONCENTRATION OF O2 [21.0]: <u>21.0</u>

Enter the O_2 concentration reading from the left display of the GMS into the computer. The reading should be about 21.0%. Do not enter the percent sign. The default value is 21.0%.

PRESS ENTER TO CONTINUE.

Press the ENTER key to begin acquiring this calibration point.

While collecting data, the computer displays:

```
COLLECTING 500 POINTS.  THIS WILL TAKE  10 SECONDS.
PLEASE WAIT PATIENTLY.
```

The calibration point for room air is now complete.

A.3.2.2 Calibration Gas - GMS Calibration

The next step in the GMS calibration requires the use of a calibration gas containing 7% CO_2, 13% O_2, balance N_2. Insert the tip of the GMS sampling capillary into a "T" connector placed in a plastic tubing connected to the bottle of compressed gas. The GMS probe should not block the flow of gas and should be at least one foot from the end of the tubing. Turn on the calibration gas and set a low flow rate. Wait for the three GMS readings to stabilize. The computer will prompt:

```
CONNECT THE GMS PROBE TO THE CALIBRATION GAS.
ENTER THE PERCENT CONCENTRATION OF CO2 [7.00]: 6.98
```

Enter the CO_2 concentration displayed on the GMS. The reading should be approximately 7%. The default value is 7.00%. This example uses an actual CO_2 concentration of 6.98%.

```
ENTER THE PERCENT CONCENTRATION OF O2 [13.0]: 13.0
```

Next, enter the O_2 concentration of the calibration gas. The reading should be approximately 13%.

```
PRESS ENTER TO CONTINUE.
```

Press the enter key to determine this calibration point. While the computer is collecting data it will display:

```
COLLECTING 500 POINTS.  THIS WILL TAKE  10 SECONDS.
PLEASE WAIT PATIENTLY.
```

After the 500 data points have been acquired, the program generates the calibration curves for both the CO_2 and O_2 channels. The GMS calibration factors are displayed on the CRT. The acutual values of the calibration factors are not important for normal operation. Press the ENTER key to return to the calibration menu.

A.3.3 Calibrating the Thermocouple

The thermocouple must be calibrated once per month. After calibrating the thermocouple, the calibration factors becomes the new default calibration.

Four water baths, with temperatures ranging from 20 ^oC to 40 ^oC, are required to calibrate the thermocouple. An automatic stirring system for the water baths should be used because a temperature gradient of several degrees exists in non-stirred water baths. Allow two hours for the water baths to reach the desired temperatures. An accurate thermometer with a range of 0 ^oC to 50 ^oC is also required.

The temperature of each bath will be measured five times, for a total of twenty measurements. Repeat the following four step sequence for each of the twenty measurements. Prompts from the computer are printed in upper case. A different bath temperature should be used for each measurement, cycling from the warmest to coolest five times.

1. Place the thermometer in the water bath to be measured. Wait for the thermometer's reading to stabilize.

2. Enter the thermometer's temperature reading into the computer at the prompt: ENTER THE TEMPERATURE OF THE WATER BATH: __

3. Place the tip of the thermocouple probe into the water bath then press the ENTER key to let the computer take a data point.

4. DO YOU WISH TO KEEP THIS DATA POINT? ([Y]/N): __
Press ENTER if the data point is valid. If you wish to ignore this data point then answer no to this question.

After repeating the above steps twenty times the computer determines a quadratic calibration curve and stores the polynomial's coefficients on the hard disk. These coefficients become the new thermocouple calibration.

A.3.4 Calibrating the Flow

There are several steps to perform during flow calibration. A team of two people is recommended, however one person can do the procedure alone. The following procedure indicates questions that will be asked by the computer in upper case characters. Typical user responses will be underlined and default answers will be enclosed in square brackets.

Flow calibration requires the following pieces of equipment. Two foam stoppers are needed that fit snugly inside the opening of the PIM. The foam stoppers are used to stop all air flow within the PIM during one part of the calibration. A metronome is required for pacing the person pumping the calibration syringe. The barometric pressure, room temperature, and relative humidity are also needed.

The questions asked during flow calibration are listed below.

```
DO YOU WISH TO USE GAS TEMPERATURE AND GAS
VISCOSITY CORRECTIONS WHILE CALIBRATING THE FLOW
SIGNAL? ([Y]/N): __
```

The default answer to this question is yes. If temperature and viscosity corrections are made then the next three questions are asked.

```
ENTER THE PERCENT RELATIVE HUMIDITY: 30
ENTER THE ROOM TEMPERATURE. (DEG F): 72
ENTER THE BAROMETRIC PRESSURE. (INCHES): 28.6
```

For each question, enter the current ambient condition. The underlined numbers represent typical responses.

```
ENTER THE ATPS PUMP VOLUME [3.06]: __
```

The ATPS (Ambient Temperature and Pressure, Saturated) pump volume is the volume in liters of the pump or syringe used for flow calibration. The volume may be determined using a small spirometer.

```
WHAT IS THE %CO2 CONCENTRATION OF THE FLOW
CALIBRATION GAS? [0.05]: __
```

Enter the percent concentration of CO_2 for the gas used for flow calibration. The default is room air.

```
WHAT IS THE %O2 CONCENTRATION OF THE FLOW
CALIBRATION GAS? [21.0]: __
```

Enter the percent concentration of O_2 for the gas used for flow calibration. The default is room air.

Because the flow transducer is not linear, flow measurements will

be made at eight different flow rates. A metronome should be used to pace the operator of the calibration syringe. Each of the eight calibration points will require the determination of binary zero flow and pumping the syringe at a specified rate. The first step is calculating binary zero flow.

```
YOU CAN DETERMINE THE BINARY ZERO FLOW OR USE A
DEFAULT VALUE OF 2036.
DO YOU WISH TO DETERMINE BINARY ZERO FLOW? ([Y]/N): __
```

Normally it is wise to determine the binary zero flow. Answer yes, the default, to determine zero flow. Place a blue stopper in each end of the face mask. The foam stoppers isolate the PTM from any room air circulation.

```
PRESS ENTER TO CONTINUE.
```

Press the ENTER key after the holes in the PTM are plugged. Binary zero flow will be determined. The zero flow obtained can range from 2026 to 2040 but will remain constant after the Godart pressure transducer has warmed up. The zero flow can be determined as often as desired.

If the thermocouple probe is not installed in the mask system, install it at this time.

```
INSERT THE THERMOCOUPLE INTO THE PTM HEAD
CONNECT PUMP FLOW TO THE PNEUMOTACHOGRAPH.

CONNECT SYRINGE TO PNEUMOTACHOGRAPH.

PUMP AT   5 CYCLES/MIN.

PRESS ENTER TO CONTINUE.
```

One person should start pumping the syringe at the indicated rate. When ready to start taking data the computer operator presses the ENTER key. The person pumping should continue until the computer beeps and displays "DATA COLLECTION COMPLETE... QUIT PUMPING.". After the data has been collected the computer integrates the flow signal and calculates inspiratory and expiratory binary volumes. The volumes will be listed in a table similar to Table A.1. The actual values listed in the table are not important, however the computer operator should keep an eye open for bad calculations. Bad binary volumes can be identified by values that are not in the same range as the other numbers in that column. Breath Number 4 of Table A.1 is an example of a bad calculation. The operator should not keep this example set of data because of the bad calculation.

	Uncorrected Values		Corrected Values	
BREATH NUMBER	BINARY AIR INSPIRED	BINARY AIR EXPIRED	BINARY AIR INSPIRED	BINARY AIR EXPIRED
1	-846.2	898.5	-680.2	726.0
2	-851.7	888.1	-684.7	717.8
3	-865.3	895.8	-696.2	725.4
4	-832.6	692.2	-669.6	612.5
5	-847.2	871.5	-681.6	705.2
6	-838.6	892.0	-676.2	722.0

Table A.1. Example listing of binary volumes for six calibration breaths.

After the binary volumes for each breath have been calculated, the program displays the flow calibration factors generated for this pumping rate. The calibration factors will only be of interest to the

user if flow calibration studies are made.

DO YOU WISH TO KEEP THIS SET OF DATA? ([Y]/N): __

If no bad volume calculations are displayed in the binary volume table then keep the set of data. If bad volume calculations are observed then answer no to this question and repeat this pumping rate.

After all eight flow rates have been completed a third-order polynomial is fit to the eight inspiratory and expiratory calibration factors. The coefficients determined by the curve fit become the flow calibration factors used by the analysis program.

A.3.5 Storing the Calibration File

The calibration file must be saved in a disk file if it is to be used for data analysis. Calibration factors may only be stored on the disk if both the GMS and flow transducers have been calibrated. If they both have not been calibrated then the user will be informed of the problem and the calibration menu will return.

If both the GMS and flow transducers are calibrated then the file name prompt shown in Figure A.4 is displayed. Enter the file name to use for the calibration file. Do not enter the volume ID of the mass storage device at this time. The "R" or "H" indicates whether room air or hot and humid air was used to calibrate the flow. This is the recommended format, however, any name with up to nine characters and numbers may be used. If more than nine characters are entered then the right-most extra characters will be deleted automatically.

```
Format:  CALmmddI
             \/\/|
             | | |
             | | |
             | | |
             | | --- R=Room, H=Heated
             | |
             | ----- Day
             |
             ------- Month
```

Enter the calibration file name: <u>CAL0205R</u>

Figure A.4. Calibration file name format.

The program needs to know which disk drive to store the calibration file on. The eight inch drive is the default disk drive for storing all disk files. It is recommended that the same eight inch disk is used to store all calibration files. Section A.1.4 describes the procedure to use for selecting a mass storage device. If an error occurs while saving the calibration file then correct the indicated problem and try saving again.

A.3.6 Return to the Main Menu

This menu selection is used to return to the main menu.

A.4 Data Collection

The data collection program is used to collect a set of respiratory data using the Data Acquisition Module (DAM). Collected data is stored on disk for later processing by the analysis program.

A.4.1 Storage Requirements

A disk with sufficient space to store the entire run must be available prior to collecting the data. For instructions on how to format a new disk refer to Appendix B, Section B.4. A simple rule for disk storage is that if the sampling frequency is 50 Hz then one eight inch disk can store up to 40 minutes of data. One disk can store more than one set of data, provided that the total time is less than or equal to 40 minutes.

If the operator is familiar with HP Pascal then the system filer can be used to determine if there is sufficient storage space on a disk. If the sample rate is fifty times per second then 24000 bytes are required per minute of data collection.

A.4.2 Operation

The data collection program is much simpler to operate than the system calibration program. The first prompt from the data collection program is listed on the following page.

```
***************************************************
*                                                 *
*    Data Acquisition Program Version 3 - DAP3    *
*                                                 *
*                                                 *
* Version: 14-Apr-87                              *
***************************************************
```

```
ENTER -1 TO EXIT TO THE MAIN MENU.
ENTER THE SAMPLING FREQUENCY [50 HZ]: __
```

Fifty Hz is the default entry for this question. If a sample rate other than fifty Hz is required then enter that value. The maximum sampling rate of the DAM is 350 Hz. If a negative sampling rate is entered then the program will return to the main menu. To select the default answer, press the ENTER key.

```
ENTER THE NUMBER OF SAMPLES TO TAKE [90000]: __
```

Enter the number of samples to be taken. If the default sample rate of 50 Hz is used, then each minute requires 3000 samples. A maximum of 90000 samples is permitted. The default entry is the maximum number of samples.

```
PRESS ENTER TO CONTINUE.
```

After entering the number of samples that are to be taken the program is ready to start collecting data. To start collecting data simply press the ENTER key. While data is being collected a display similar to the following is shown on the CRT.

```
NOW COLLECTING DATA... please wait patiently.
Collecting 27000 samples. This will take 540 Sec
```

After the data is collected, a list of the minimums, maximums, and the number of out of range samples is displayed on the CRT. Figure A.5 represents an example of the minimum and maximum listing. If any of the signals are out of range then the DAM should be adjusted.

```
DATA COLLECTION COMPLETE.
Calculating the maximum and minimum values...
Signal     Max   Min        #4095 s     #0 s
 CO2      3128   127          0          0
  O2      2839   521          0          0
 FLOW     2654   302          0          0
 Temp     2470   817          0          0
```

Figure A.5. Example of minimum and maximum sample values.

A.4.3 <u>Storing the Data</u>

If all of the values are within range then the data is good and should be stored. If there are values out of range then the data should be stored anyway but the operator should be aware that the analysis of the data may be in error. To proceed to the store data prompts press the ENTER key.

```
Format:  mmddIwww
         \/\/|\_/
         | | | |
         | | | --- Work Load
         | | |
         | | ----- Subject Identifier
         | |
         | ------- Day
         |
         --------- Month

Enter the file name: __
```

Figure A.6. Recommended file name format for binary data.

Figure A.6 shows the recommended file name format for storing the binary data. Any name may be used so long as the length is eight or fewer characters and only letters and numbers are used. If the length is greater than eight characters then it will be shortened automatically. For example, if the date is January 6, the subject is Gary and the work load is 75 watts then the file name would be "0106G075".

The volume ID of the desired mass storage device will be selected next. See Section A.1.4 for information on selecting the correct mass storage device. Storing the data may require up to four minutes. The program returns to the calibration menu after storing the data.

A.5 Analyzing the Binary Data

The analysis program combines calibration information with binary respiratory data to calculate the respiratory parameters. An entire set of respiratory data or only a window may be analyzed. Windowing is used to calculate the respiratory parameters during a short time period, emulating the Douglas bag technique. Figure A.7 shows the analysis program menu.

```
**********************************************
*                                            *
*              ANALYSIS MENU                 *
*                            Current file    *
*  1) Load CALIBRATION.       CAL1015R       *
*  2) Load BINARY DATA.       1025G075       *
*  3) Plot binary data.                      *
*  4) Change current defaults.               *
*  5) Analyze the data.                      *
*                                            *
*  6) Return to the MAIN MENU                *
*                                            *
*                                            *
*  Enter your choice: __                     *
*                                            *
*                                            *
* Version:  14-Apr-87                        *
**********************************************
```

Figure A.7. Menu for the analysis program.

Not all six of the menu selections are displayed when the analysis program executed. is As data and calibration files are loaded and analyzed the other menu functions become available. The current file heading of the analysis menu shows the file names of the binary data and calibration factors that are currently loaded. Descriptions of the six menu selections are located in the following sections.

A.5.1 Load Calibration

This menu selection is used to load a calibration file into memory. Section A.3 describes the system calibration procedure. Figure A.8 shows the recommended file name format for calibration files.

```
Format:  CALmmddI
            \/\/|
            | | |
            | | |
            | | |
            | | -- R=Room, H=Heated
            | |
            | ---- Day
            |
            ------ Month

ENTER THE CALIBRATION FILE NAME: _
```

Figure A.8. The recommended calibration file name format.

This is the same file name format recommended by the calibration program.

The mass storage device that contains the calibration file must be determined. Section A.1.4 describes the method used to select a mass storage device. Enter the number corresponding to the disk drive that contains the calibration file. The default mass storage device is selection number three, the eight inch floppy disk drive. If the file is not found then an error message is displayed.

A.5.2 Load Binary Data

Binary data is the sampled data collected from a subject by the data collection program. Figure A.9 shows a recommended file name

format. This is the same format that was recommended for storing the data in Section A.4.3.

```
Format:  mmddIwww
         \/\/|\_/
         | | | |
         | | | --- Work Load
         | | |
         | | ----- Subject Identifier
         | |
         | ------- Day
         |
         --------- Month

Enter the file name: _
```

Figure A.9. Recommended binary data file name format.

The mass storage device that contains the calibration file must be determined. Section A.1.4 describes the method used to select a mass storage device. Enter the number corresponding to the disk drive that contains the binary data file. The default mass storage device is selection number three, the eight inch floppy disk drive. If the file is not found then an error message is displayed. Correct the problem and try again.

ENTER SUBJECT'S NAME OR IDENTIFIER.
DEFAULT: default name

The subject's name is used by the program to retrieve the age, height, and weight of the subject from a file. Additional names can be added to the list by using the Edit Subject Information selection of the main menu. Section A.7.2 describes adding names to the list.

A.5.3 Plot the Binary Data

This function is only available when binary data is loaded into memory. The binary data can be plotted on either the CRT or the plotter. If a calibration file is also loaded into memory then the units of the data will be displayed. Otherwise, the units used for the plot will be the binary values returned from the DAM.

ENTER THE FIRST POINT TO PLOT [1]: __

Enter the first point of the plot. Any value may be selected as long as it is within the range of the binary data. The default value is the first data point in memory.

ENTER THE LAST POINT TO PLOT: __

Enter the last point to be plotted. Any value may be selected as long as it is greater than the first point and is less than or equal to the maximum number of points in memory. The last point in memory is the default answer. If more than 3000 points are selected then expect to wait several minutes for the plot be drawn.

ENTER THE PLOT DEVICE. (C=CRT,P=PLOTTER)[C]: __

Enter "C" if the plot is to be on the computer's CRT or "P" if the plotter should be used. The CRT is the default entry.

After the plot is finished the computer beeps and displays the message:

DO YOU WISH TO REDO THE PLOT? (Y/[N]): __

Entering "yes" starts the plotting routine again. "No" returns to the analysis menu. Figure A.10 shows an example plot of 2000 points.

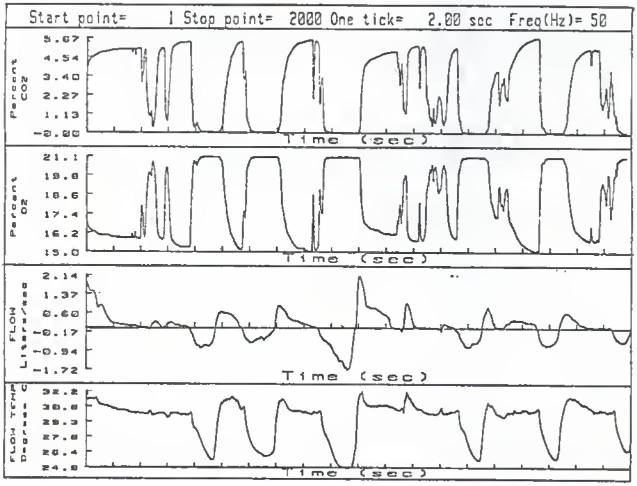


Figure A.10. Example plot of 2000 respiratory data points.

A.5.4 Change Current Defaults

Many default parameters can be modified by the user before the analysis is run. All of the parameters are set to default values prior to analyzing the respiratory data. The following is a list of

the questions and how they should be answered.

ENTER SUBJECT'S NAME OR IDENTIFIER.
DEFAULT: default name

The subject's name is used by the program to load the subject's age, height, and weight of the subject from a file. Additional names can be added to the list by using the Edit Subject Information selection of the main menu. Section A.7.2 describes adding names to the list.

ENTER THE BEGINNING TIME OF DATA
COLLECTION. (SEC) [0]:

The beginning time of collection is not used by the analysis program. This value is passed on to the display program as a time reference for generating plots. If one minute of heart rate is collected before starting data collection then this value can be set to 60 seconds. The default time is zero seconds.

ENTER A COMMENT TO BE PRINTED ON THE
HARD COPY OUTPUT. (60 CHARS MAX)

This requests a line of text that will be printed with the analyzed results. The default is a blank line.

ENTER SAMPLING FREQUENCY [50]: __

Enter the sampling frequency used for data collection. The default is 50 samples per second.

B-BY-B TIME DELAY OR FIXED? ([B]/F)

From 380 to 700 milliseconds are required for the gas sample to reach the mass spectrometer. The breath-by-breath method of delay calculation determines the GMS time delay for each breath. A fixed time delay uses a constant time delay for the entire analysis. Breath-by-breath calculation of the time delay is the preferred method and is also the default.

If breath-by-breath is used then the following question is asked.

```
ENTER AVE TIME DELAY FOR BAD BREATH PROBLEMS
[500]: __
```

The value entered is used if there are problems calculating a time delay for a breath.

If a fixed GMS time delay is selected then the following question is asked.

```
ENTER THE NEW TIME DELAY: __
```

Enter the time in milliseconds that is to be used for the constant GMS time delay.

```
USE TEMP AND VISCOSITY CORRECTIONS? ([Y]/N): __
```

Temperature corrections are used to convert between BTPS and STPD conditions. Viscosity corrections adjust measured volumes for viscosity changes as a function of the CO_2, O_2, and N_2 concentrations and gas temperature. If temperature and viscosity corrections are used then the next two questions are asked.

```
ENTER THE BODY TEMPERATURE (DEG F) [98.6]: __
```

The body temperature is used to convert from STPD to BTPS conditions. The normal procedure is to use the end-exercise body temperature.

```
BAROMETRIC PRESSURE = 40.0
RELATIVE HUMIDITY   = 40.0
ROOM TEMPERATURE    = 74.0

DO YOU WISH TO CHANGE THE ABOVE AMBIENT
CONDITIONS? (Y/[N]): __
```

These are the ambient conditions that were present during system calibration. If the ambient conditions were different during data collection then answer yes to change them.

```
DO YOU WISH TO PRINT THE BREATH BY BREATH OUTPUT
OF THE RESULTS AS THE DATA IS BEING ANALYZED?
(Y/[N]): __
```

Answer yes if the respiratory parameters are to be printed for each breath in addition to the averaged parameters that are normally generated.

A.5.5 Analyze Data

Both a calibration file and binary data must be loaded into memory before the analysis can begin.

```
ENTER STARTING POINT TO ANALYZE [1]: __
ENTER ENDING POINT TO ANALYZE [default value]: __
```

The operator is required to enter the starting and ending data

points that are to be included in the analysis. The default values are the first and last data points in memory. For example, if the sampling rate is 50 Hz and the time window to be analyzed is from five minutes, fifteen seconds (5:15) to six minutes, zero seconds (6:00) then the starting point for the analysis is 15750 and the ending point is 18000.

```
DO YOU WISH TO STORE THE ANALYZED RESULTS IN
A DISK FILE? ([Y]/N): __
```

There are two forms of output available. The first is a printout of the average respiratory parameters and calibration information. The second form of output is a disk file containing the analyzed data for each breath. The second output type is optional. This question is used to determine if the results of the analysis should be stored in a disk file. The results must be stored if the results are to be plotted by the display program. The default answer to the question is yes. A recommended file name format for storing the results is shown in Figure A.11. This is the same file name format used to store the binary data. The program automatically precedes the file name with an identifier so the binary data is not accidentally erased.

After the binary data are analyzed, the results are printed on the Decwriter II. The analysis menu shown in Figure A.7 will return after the printing of the analyzed results is finished.

```
Format:  mmddIwww
         \/\/|\_/
         | | | |
         | | | --- Work Load
         | | |
         | | ----- Subject Identifier
         | |
         | ------- Day
         |
         --------- Month

Enter the file name [default name]: _
```

Figure A.11. Recommended file name format for storing analyzed data for later plotting.

A.5.6 Return to the Main Menu

This menu selection is used to exit the analysis program and return to the main menu of the respiratory analysis system. It is selected by entering a "6" from the analysis menu which is displayed in Figure A.7.

A.6 Plotting and Displaying Analyzed Results

The display program is used to do additional analysis and plotting of previously analyzed runs. The results can be printed in tabular form or plotted on either the CRT or plotter. If more than one set of results are averaged then the standard deviation can also plotted or printed.

Before the plotting and printing options are described there are several features that must be described. Each of the following features are used by one or more of the plotting options.

A.6.1 General Information

A.6.1.1 Loading Analyzed Data from Disk

If the analyzed results have already been loaded into memory then it is not necessary to reload the results again. The following questions are asked each time that new results can be loaded into memory. If any results are already in memory then the question:

```
DO YOU WISH TO USE THE SAME DATA? ([Y]/N): __
```

is displayed. Answer yes if the results currently in memory are to be retained. Answer no if new results should be loaded from disk.

If menu selection seven, plot averaged data, is selected then the two questions:

```
HOW MANY RUNS WOULD YOU LIKE TO AVERAGE
TOGETHER FOR EACH PLOT? [1]: _3_

HOW MANY AVERAGED PLOTS DO YOU WANT ON EACH
GRAPH? [1]: _2_
```

are displayed. For the example of Section A.6.4.7 that plotted two work loads, with each work load being the average of three runs, the user would enter "3" for the number of runs averaged for each plot and "2" for the number of plots to be drawn.

If any menu selection other than two, four, or seven is selected then the question

```
HOW MANY RUNS WOULD YOU LIKE TO EXAMINE
SIMULTANEOUSLY? [1]: __
```

is displayed. Enter the number of runs that will be plotted, or combined to form one plot and its standard deviation.

Next, enter the names of all of the data files to be loaded. The format of Figure A.12 is recommended. This is the same format recommended in the analysis program for storing the analyzed results.

```
Format:  mmddIwww
         \/\/|\_/
         | | | |
         | | | --- Work Load
         | | |
         | | ----- Subject Identifier
         | |
         | ------- Day
         |
         --------- Month

Enter the file name: __
```

Figure A.12. Recommended file name format for loading analyzed results.

The file name format is only a recommendation. Any name may be used as long at there are no more than eight characters. If more than eight characters are entered then only the first eight are used.

After loading all of the necessary files, the program advances to questions concerning the averaging technique to be used and whether bad breaths are to be omitted.

A.6.1.2 Averaging Techniques

Three methods of combining the analyzed breath-by-breath results are available. The three methods are listed in Figure A.13.

```
How do you wish to combine the breaths?

    1. The individual breaths.
    2. Average individual breaths.
    3. Average windows of breaths.

Enter your choice (1,2,[3]): _
```

Figure A.13. The three averaging techniques that can be used to filter the respiratory data.

In the first method, each breath is treated as a unique data point. Thus, no filtering is done to the results. This selection should not be used with the polynomial curve-fitting option because a maximum of 300 data points are permitted.

In the second method, the user selects the number of breaths that will be averaged to form one data point. Five breaths is a typical number of breaths to average. Method three uses a sliding time window. The user specifies the beginning and ending time for the plot. A window period and window width must are also required. The window width specifies the number of seconds included in each window. The window period specifies how far the window should be slided to position the next window. Figure A.14 illustrates the sliding time

window technique. Typical window parameters are a period of five seconds and a width of ten seconds.

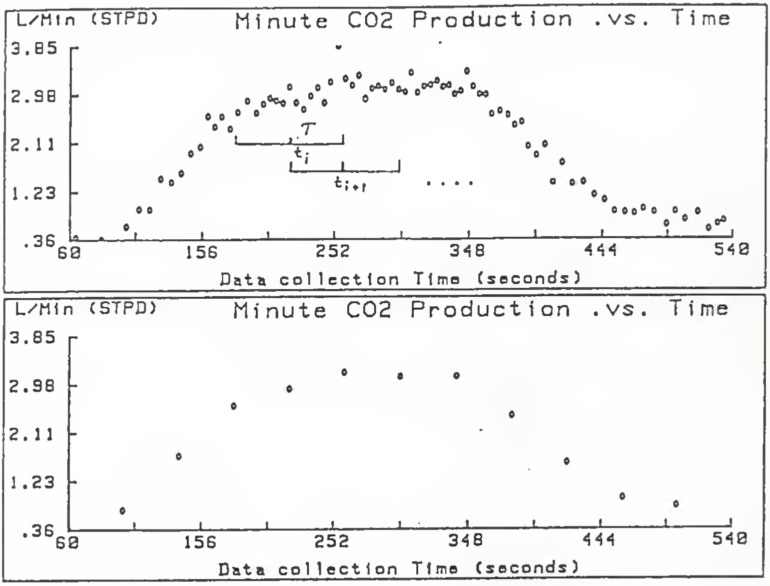


Figure A.14. Illustration of the sliding window averaging technique. Reproduced from Masters (3).

A.6.1.3 <u>Bad Breaths</u>

If either the subject's inspirate or expirate volume is less than 0.4 liters then the breath is flagged as bad. Normally bad breaths are omitted from all calculations and plots, however, the user may include them if desired.

A.6.1.4 Special Questions for Plots

The three plotting routines require more information than the printing routines. The additional questions are described in this section.

A.6.1.4.1 Selecting Parameters to Plot

Up to two respiratory parameters can be plotted at one time. If only one parameter is to be plotted then just press the ENTER key when the number of the second plot is requested. The items to be plotted are chosen by the menu shown in Figure A.15. If the variable vs. variable plotting option is chosen then two parameters must by selected. The first parameter is plotted on the independent axis and the second parameter is plotted on the dependent axis.

```
Which of the following respiratory variables
do you wish to plot?  Enter the number
corresponding to the parameters chosen.  Up
to two may be selected.

    1.  Oxygen consumed
    2.  Carbon dioxide produced
    3.  Alveolar oxygen consumption
    4.  Alveolar carbon dioxide production
    5.  Inspiratory Minute ventilation
    6.  Expiratory Minute ventilation
    7.  Inspiratory Tidal volume
    8.  Expiratory Tidal volume
    9.  Lung volume
   10.  Respiratory Frequency
   11.  Respiratory Quotient
   12.  Alveolar Respiratory Quotient

Enter the first plot: _3_
Enter the second plot (0=no plot): _4_
```

Figure A.15. The parameters that can be plotted.

A.6.1.4.2 Using Either the CRT or Plotter

Plots can be made either on the CRT display of the computer or the eight-pen vector plotter. The default plotting device is the CRT. The following question is asked when it is time to choose the plotting device:

OUTPUT TO PLOTTER OR CRT (P/[C]): __

If the CRT is chosen as the plotting device then the maximum and minimums calculations are performed. The maximum and minimum calculations are described in Section A.6.1.4.3.

If the plotter is chosen as the plotting device then the following set of questions are asked.

DO YOU WISH TO DRAW THE AXES ([Y]/N): __

Normally the axes should be drawn. The only time that a "no" response is appropriate is when several plots are made on top of each other. If the axes are drawn each time that a new plot is added to the pile then considerable time is wasted and the ink on the axes labels becomes smudged.

There are eight pens that can be used by the plotter. It is possible to draw each plot in a different color. The questions listed in Figure A.16 are used to select which pen to use when drawing multiple graphs. Figure A.16 represents an example where the axes and plot labels are drawn using pen number one, the first plot with pen number two, and plot number two is drawn with pen number three. The

user must ensure that pens are properly installed in the plotter. Pen number one is always used for the axes and plot labels.

```
You will now need to enter the pen number that
will be used for each curve.

ENTER the pen number for curve number 1 [1]: _2_
ENTER the pen number for curve number 2 [2]: _3_
```

Figure A.16. Example of selecting plotter pens.

Be sure that plotter paper is positioned correctly on the surface of the plotter before continuing.

A.6.1.4.3 Maximum and Minimum Values

The maximum and minimum values of the dependant axis can be specified by the user or permitted to default. The questions that prompts for the axis limits are similar to the following:

```
ENTER THE MIN VALUE FOR ALVEOLAR OXYGEN CONSUMPTION
OR ENTER FOR CALCULATED MIN.
```

The parameter that is listed will depend on the answer to the question of Section A.6.1.4.1. Either type a value for the minimum or maximum axis value or press the ENTER key. If just the ENTER key is pressed then the program will determine appropriate maximum and minimum values.

A.6.1.4.4 Drawing the Plot

After the information from the previous sections is entered into the computer, the plot is drawn on the selected device. Plotting on the CRT requires only a few seconds while the plotter requires several

minutes. If the CRT is used then the ALPHA and GRAPHICS key can be used to control the display of either the text display, graphics display, or both. The default is for both displays to be superimposed on the same screen. Pressing either the ALPHA or GRAPHICS key twice selects that display as the new default. Pressing the SHIFT and the GRAPHICS keys simultaneously will copy the graphic image to the thermal printer.

A.6.1.4.5 Labeling the Plot

If more than one set of data is plotted on one set of axes then it is advisable to identify the source of each curve. For example, the identifier "xxx - 150 WATT WORKLOAD" of Figure A.18 was entered in response to the question:

```
ENTER THE DESCRIPTION FOR PLOT #  1
OR PRESS <ENTER> TO LEAVE BLANK.
```

The preceding question is asked for each set of respiratory data plotted on the axes. Up to six data sets can be identified using this technique. If the ENTER key is pressed without typing an identifier then no label will be drawn.

```
ENTER A TITLE FOR THIS PLOT.  THE TITLE WILL BE
CENTERED AT THE BOTTOM OF THE PLOT.
```

The plot title is the final item drawn. The title is centered at the bottom of the plot. If the ENTER key is pressed without typing an identifier then no plot title is drawn.

A.6.2 Display Menu

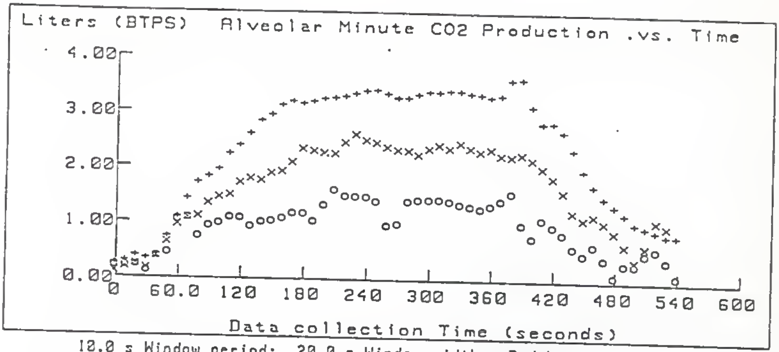
The display menu is shown in Figure A.17. Descriptions of the nine menu choices are found in Sections A.6.2.1 through A.6.2.9.

```
*******************************************
*                                         *
*              Display Menu               *
*                                         *
*   1) Plot results.                      *
*   2) Print results.                     *
*   3) Print the averaged results.        *
*   4) Plot variable vs. variable.        *
*   5) Curve fit the results.             *
*   6) Curve fit the averaged results.    *
*   7) Plot the averaged data.            *
*   8) HELP with menu selections.         *
*                                         *
*   9) Return to MAIN MENU                *
*                                         *
*  Enter your choice: __                  *
*                                         *
* Version:  14-Apr-87                     *
*******************************************
```

Figure A.17. Display program menu.

A.6.2.1 Plot the Results

This selection is used to plot the analyzed results vs. time on either the CRT or plotter. Up to eleven runs may be stored in memory at one time. If the average of more than one run is requested then plus and minus one standard deviation is also plotted. Otherwise, each run can be plotted using a different color. All three of the averaging techniques can be used to filter the results. Two example plots created using this menu choice are shown in Figures A.18 and A.19. Figure A.18 shows three sets of data and Figure A.19 shows the average and standard deviation of the same data. The averaging used a windowing period of five seconds and window width of ten seconds.

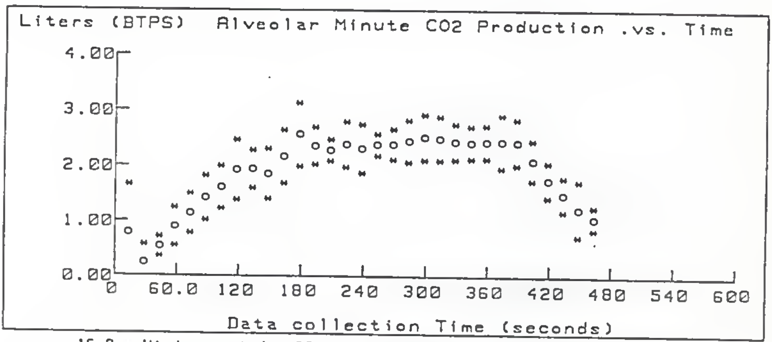


10.0 s Window period; 20.0 s Window width; Bad breaths omitted

OOO - 75 WATT +++ - 200 WATT
XXX - 150 WATT

Subject S

Figure A.18. Example of plotting three sets of respiratory data on one set of axes.



15.0 s Window period; 20.0 s Window width; Bad breaths omitted

Figure A.19. Example showing the average and plus/minus one standard deviation of the respiratory data shown in Figure A.18.

A.6.2.2 Print the Results

This selection is used to print the results of one run on the Decwriter II printer. Any of the three averaging techniques can be used. Table A.2 shows typical results generated by this selection. Once again a window averaging technique is used. Both the window period and window width are thirty seconds.

A.6.2.3 Print the Averaged Results

This selection is used to print the average of more than one run on the printer. Only the windowing averaging technique is can be used. Table A.3 shows an example of printing the average and standard deviations of three sets of data.

A.6.2.4 Plot Variable vs. Variable

This menu selection is used to plot two respiratory parameters against each other. Figure A.20 shows an example plot of oxygen consumption vs. carbon dioxide production. Any of the three averaging techniques can be used to filter the data before plotting.

A.6.2.5 Curve Fit the Results

This selection is used to least-squares fit a polynomial to the results of a run. Up to a sixth order polynomial may be fit to the data, with a maximum of 150 points used in the fit. Because of the limited number of points, a window width of less than fifteen seconds is not recommended. More than one curve may be plotted at one time with each curve representing a different set of respiratory data.

Date: 11/20/86
Comment:
Subject identifier: JAMES

The window period is 30.0 seconds
The window width is 30.0 seconds
The starting time is 0.0 seconds
The ending time is 540.0 seconds

The breaths are combined by WINDOWING. The breaths occurring in a window 30.0 second(s) wide are averaged. The center of the window is moved 30.0 seconds between successive averages. The center of the first window is 0.0 seconds and the ending time is 540.0 seconds.

* BAD BREATHS OMITTED *
The bad breaths are not included in the averaging technique. Bad breaths are those breaths with an inspiratory volume or an expiratory volume of less than 0.4 liters.

The R' value is equal to the averaged CO2 produced divided by the averaged O2 consumed. It is a better estimate of the true R' value for the breaths averaged than the average of the individual R' values.

Time (sec)	O2 Consumed (L/min) (STPD)	CO2 Produced (L/min) (STPD)	Alveolar O2 Consumed (L/min) (STPD)	Alveolar CO2 Produced (L/min) (STPD)	Resp. Quotient (STPD)	FRC Resp. Quotient	Averaged CO2/O2 :R'	Ventilation Insp. (L/min) (BTPS)	Ventilation Expr. (L/min) (BTPS)	Tidal Volume Insp. (L) (BTPS)	Tidal Volume Expr. (L) (BTPS)	Resp. Freq. (br/min)
0.0	0.185	0.160	0.317	0.213	0.864	0.634	0.863	11.91	12.09	0.72	0.73	16.54
30.0	0.304	0.369	0.355	0.300	0.817	0.834	1.214	13.56	14.42	0.75	0.81	17.86
60.0	1.000	1.007	1.144	0.988	1.283	0.890	1.007	33.48	34.47	1.16	1.20	28.25
90.0	1.618	1.331	1.761	1.327	0.904	0.755	0.823	38.40	38.51	1.26	1.26	30.99
120.0	2.018	1.680	1.955	1.708	0.921	0.878	0.833	47.76	47.18	1.62	1.61	29.46
150.0	2.242	1.833	2.024	1.941	0.847	0.964	0.818	52.24	50.25	1.76	1.70	29.68
180.0	2.250	2.205	2.101	2.258	1.030	1.131	0.980	62.45	61.61	2.12	2.09	29.62
210.0	2.331	2.342	2.135	2.380	1.472	1.116	1.005	68.96	67.98	2.32	2.29	29.74
240.0	2.673	2.382	2.301	2.558	0.955	1.114	0.891	70.62	67.66	2.38	2.28	29.65
270.0	2.612	2.246	2.279	2.386	0.922	1.052	0.860	65.32	62.66	2.20	2.11	29.71
300.0	2.595	2.246	2.333	2.383	1.087	1.099	0.866	65.22	62.85	2.21	2.12	29.68
330.0	2.482	2.269	2.206	2.340	0.958	1.065	0.914	67.22	65.43	2.27	2.20	29.67
360.0	2.639	2.286	2.384	2.433	1.173	1.027	0.866	66.77	64.35	2.25	2.17	29.68
390.0	2.714	2.121	2.266	2.276	0.839	1.009	0.781	63.29	59.75	2.15	2.03	29.46
420.0	1.918	1.741	1.649	1.855	0.972	1.146	0.908	49.10	47.37	1.71	1.65	28.74
450.0	0.798	1.285	0.802	1.223	1.967	1.572	1.609	37.17	38.29	1.64	1.70	24.18
480.0	0.594	0.830	0.489	0.798	2.182	1.620	1.498	32.78	32.93	1.62	1.62	20.82
510.0	0.544	0.672	0.632	1.030	2.993	1.972	1.236	29.41	28.94	1.63	1.61	18.22
540.0	0.140	0.238	0.202	0.201	9.291	1.005	1.704	15.02	15.29	1.00	1.01	15.02

Table A.2. Typical result generated by printing a set of results.

```
Cosment:
Subject identifier: JAMES
The window period is:   30.0 seconds
The window width is :   30.0 seconds
The starting time is:    0.0 seconds
The ending time is  :  540.0 seconds

The breaths are combined by WINDOWING.  The breaths occurring in a window   30.0 second(s) wide are aversged.
The center of the window is moved   30.0 seconds between successive averages.  The center of the
First window is    0.0 seconds and the ending time is  540.0 seconds.

Bad breaths were NOT omitted.
```

AVERAGE RESPIRATORY DATA FOR 3 RUNS
ALONG WITH STANDARD DEVIATION

Time (sec)	O2 Consused (L/min) (STPD)	CO2 Produced (L/min) (STPD)	Alveolar O2 Consumed (L/min) (STPD)	Alveolar CO2 Produced (L/min) (STPD)	Ventilation Insp. (L/min) (BTPS)	Ventilation Expr. (l/min) (BTPS)	Tidal Voluse Insp. (L) (BTPS)	Tidal Voluse Expr. (L) (BTPS)	Resp. Freq. (br/min)	FRC Resp. Quotient
30.0	0.421+0.44	0.332+0.24	0.520+0.50	0.416+0.43	13.83+1.70	14.05+1.35	0.80+0.36	0.87+0.39	21.306+2.91	0.83+0.30
60.0	1.026+0.45	0.937+0.27	1.214+0.30	0.921+0.26	30.08+2.06	30.57+1.98	1.30+0.54	1.32+0.54	30.490+2.08	0.78+0.36
90.0	1.807+0.43	1.355+0.15	1.842+0.28	1.413+0.29	37.43+1.26	36.62+1.47	1.74+0.83	1.72+0.79	27.236+3.49	0.76+0.08
120.0	1.990+0.47	1.815+0.37	2.035+0.29	1.803+0.31	45.98+1.61	46.35+0.91	2.19+0.97	2.22+0.92	23.287+2.89	0.89+0.19
150.0	2.304+0.30	1.899+0.64	2.055+0.36	1.998+0.54	55.80+2.48	53.69+2.87	2.08+0.56	1.99+0.58	28.030+1.28	0.98+0.33
180.0	2.515+0.52	2.228+0.39	2.281+0.42	2.379+0.40	63.40+1.71	61.36+1.54	2.14+0.31	2.05+0.34	30.061+0.87	1.07+0.24
210.0	2.591+0.52	2.268+0.27	2.310+0.42	2.393+0.25	65.07+1.97	62.69+2.30	2.20+0.35	2.12+0.41	29.648+0.31	1.04+0.27
240.0	2.490+0.75	2.332+0.31	2.283+0.40	2.411+0.41	64.91+2.39	63.41+2.06	2.23+0.40	2.18+0.38	29.224+0.86	1.09+0.30
270.0	2.701+0.40	2.276+0.32	2.348+0.26	2.433+0.27	65.24+0.78	62.30+1.34	2.19+0.13	2.09+0.23	29.827+0.45	1.04+0.16
300.0	2.725+0.42	2.427+0.43	2.410+0.28	2.555+0.41	69.55+2.43	67.11+2.26	2.35+0.46	2.26+0.43	29.709+0.43	1.09+0.17
330.0	2.627+0.41	2.337+0.35	2.342+0.37	2.450+0.33	67.91+1.76	65.65+1.30	2.28+0.34	2.21+0.26	29.746+0.45	1.05+0.14
360.0	2.726+0.31	2.367+0.33	2.409+0.18	2.502+0.26	69.49+1.65	66.87+1.59	2.34+0.30	2.26+0.29	29.691+0.20	1.04+0.13
390.0	2.726+0.32	2.270+0.44	2.356+0.30	2.406+0.39	65.89+1.61	62.92+1.78	2.23+0.27	2.12+0.30	29.631+0.61	1.03+0.19
420.0	1.872+0.46	1.708+0.38	1.643+0.11	1.808+0.29	49.57+1.73	48.03+2.18	1.74+0.37	1.69+0.44	28.455+0.68	1.12+0.22
450.0	0.944+0.46	1.257+0.51	0.854+0.33	1.238+0.50	38.39+2.86	38.67+2.78	1.58+0.55	1.59+0.58	27.687+2.11	1.54+0.40
480.0	0.760+0.42	0.935+0.26	0.596+0.33	0.910+0.33	35.33+1.60	34.90+1.40	1.42+0.47	1.40+0.47	25.770+2.22	1.69+0.54
510.0	0.373+0.35	0.529+0.38	0.451+0.23	0.601+0.42	22.55+2.14	22.73+2.32	1.08+0.43	1.08+0.45	24.223+2.19	1.43+0.68
540.0	0.276+0.64	0.310+0.62	0.228+0.50	0.271+0.58	14.87+3.86	14.73+3.84	0.69+0.83	0.71+0.84	14.653+3.83	0.74+0.86

Table A.3. Typical results generted by averaging three runs.

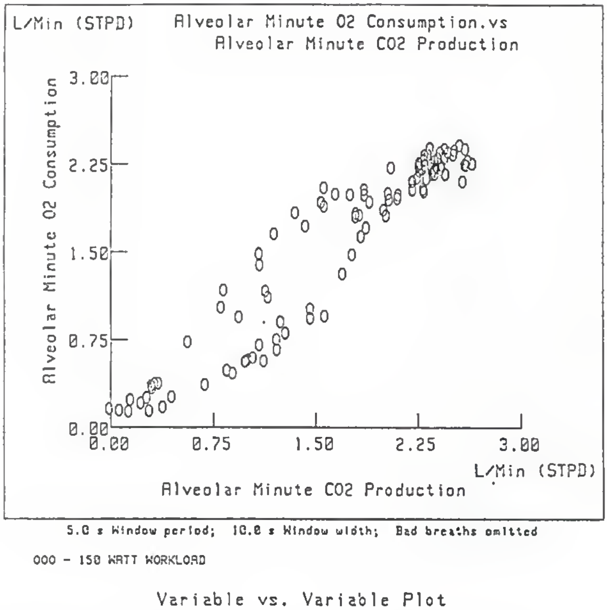


Figure A.20. Example of a variable vs. variable plot.

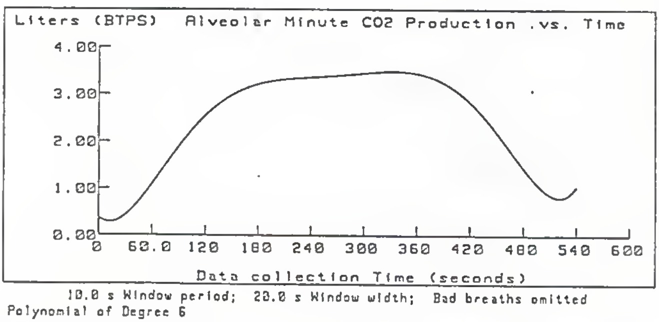


Figure A.21. Example of a sixth order polynomial fitted to a set of 150 watt exercise data.

A.6.2.6 Curve Fit the Averaged Results

This selection is similar to number five, which is described in Section A.6.2.5, except that more than one set of data can be combined into a single least-squares fit polynomial. Only the window averaging technique may be used for filtering the results. An example plot is not included because the result is similar to a curve fit with only one set of data.

A.6.2.7 Plot the Averaged Data

This is the most useful of the plotting routines. Only time window can be used to filter the results. A typical application of this routine is if three runs are performed at two different workloads then this routine would draw two plots. Each plot would be the average of three runs at the same work load. Figure A.22 shows a plot that made using this example.

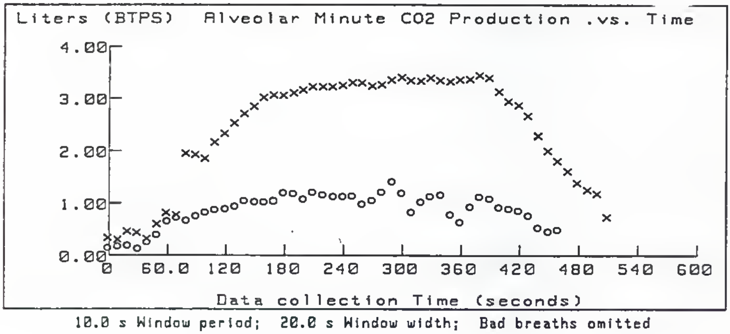


Figure A.22. Example of averaging six runs together to form two composit plots.

A.6.2.8 Help with Menu Selections

This selection displays an abbreviated set of these instructions on the computer's screen.

A.6.2.9 Return to MAIN MENU

This menu selection returns control to the main menu program.

A.7 Subject Information

The age, height, and weight of the subject are used by the analysis program to make accurate functional residual capacity (FRC) corrections. Editing the subject information is used to maintain a table containing the age, height, and weight of all of the subjects that are being studied. The subject information menu is shown in Figure A.23. The four menu choices are now described.

```
*********************************************
*                                           *
*       Edit Subject Information Menu       *
*                                           *
*  1) Print subject information on printer  *
*  2) Add a subject                         *
*  3) Delete a subject                      *
*                                           *
*  4) Return to MAIN MENU                   *
*                                           *
*  Enter your choice: __                    *
*                                           *
* Version:  9-Feb-87                        *
*********************************************
```

Figure A.23. Edit subject information menu.

A.7.1 Print Subject Information on Printer

This function prints the list of all of the subjects currently stored by the system. Table A.4 shows an example printout. The printout is made on the thermal printer so permanent records are easy to maintain. The subject number is used for deleting subjects from the table.

Number	Name	Age (yrs)	Height (inches)	Weight (lbs)
1	GARY	23	68	130
2	STEVEN	20	70	148
3	BRAD	25	71	188
4	JAMES	23	69	160

Table A.4. Example of subject information printout.

A.7.2 Add a Subject

The prompts listed in Figure A.24 are displayed when adding a subject to the list. If no subjects need to be added then just press the ENTER key for the subject's name.

```
ENTER THE FOLLOWING INFORMATION.

              NAME: _
       AGE (YEARS): _
   HEIGHT (INCHES): _
      WEIGHT (LBS): _
```

Figure A.24. Prompts used to add a subject to the subject list.

After all four questions are answered the updated file is stored on disk. The new subject is added to the end of the subject list.

A.7.3 Delete a Subject

The following prompt is displayed when deleting a subject from the list of subjects.

```
ENTER THE SUBJECT NUMBER TO BE DELETED: _
```

Enter the number of the subject to be deleted. The subject's number is found by printing the subject information on the printer.

After a subject is deleted, the subject number of all subjects listed after that subject are decreased by one. If an invalid subject number is entered then no change is made to the table.

A.7.4 Return to MAIN MENU

This function returns to the main menu program.

A.8 Set the Date

All files are stamped with a date marker whenever they are modified. This feature of HP Pascal is only useful if the user has set the date. When asked to enter the current date use the format MM/DD/YY. For example: If the date is January 5, 1987 then enter 01/05/87 for the date. Pressing ENTER without entering a date leaves the system date unchanged.

After the system date is set the main menu is displayed on the screen.

A.9 Scan for Files

The scan for files menu selection is used to search a disk for possible respiratory data or calibration files. Because each type of file has a unique file name prefix, this program can search directory listings for the prefix. Figure A.25 shows the menu that is displayed when this option is selected from the main menu.

```
*********************************************
*                                           *
*              Scan Disk Menu               *
*                                           *
* This program allows you to scan a disk    *
*   for calibration files, binary data, and *
*   analyzed data.                          *
*                                           *
* Results are displayed on the printer.     *
*                                           *
* Which type of file to scan for?           *
*                                           *
*   1) Calibration                          *
*   2) BINARY data                          *
*   3) Analyzed respiratory data            *
*                                           *
*   4) Exit to MAIN MENU                    *
*                                           *
*  Enter your choice: __                    *
*                                           *
* Version:  24-Feb-87                       *
*********************************************
```

Figure A.25. Scan disk menu.

A.9.1 General Information

After selecting the data file type the mass storage device must be selected. Refer to Section A.1.4 for instructions on choosing a mass storage device.

The search is made by executing the operating system's filer program. Because the filer is used to make the listing, there is no

way to interpret the results before printing. Therefore, the user must strip the prefix characters from the file names of all data files. After the listing is made on the printer the program will return to the main menu of this anaysis system.

A.9.2 Calibration Files

When calibration files are found on a disk, a listing similar to the one shown in Figure A.26 is made. All of the calibration files are stored in files of type ASCII. Do not include the .ASC sufix when entering the calibration file names into programs. Figure A.26 shows that the two calibrtion files CAL1106R and CAL1107R are on the selected disk.

```
-------------------------------------------------------------
          Calibration example
-------------------------------------------------------------

B9826:             Directory type= LIF level 0
created                    block size=256
 Storage order
...file name...     # blks    # bytes  last chng

CAL1106R    .ASC       2          512  6-Nov-86
CAL1107R    .ASC       2          512  7-Nov-86
FILES shown=2 allocated=23 unallocated=441
```

Figure A.26. Example directory listing showing the two calibration files CAL1106R and CAL1107R.

A.9.3 BINARY Data Files

When unanalyzed, binary data is stored on a disk then a listing similar to the one shown in Figure A.27 is made. Be sure to ignore the "F" prefix when entering file names. The "F" is added to the file name automatically. Figure A.27 indicates that there are three files

on the selected disk. The file names are 1103B150, 1105B150, and 1111B075.

```
------------------------------------------------------------
  The following files may contain valid BINARY data.
  Ignore the leading F when entering file names.
------------------------------------------------------------

B9826:               Directory type= LIF level 0
created                      block size=256
 Storage order
...file name...       # blks     # bytes  last chng

F1103B150                211       54004  3-Nov-86
F1105B150                211       54004  5-Nov-86
F1111B075                211       54004 11-Nov-86
FILES shown=3 allocated=28 unallocated=441
```

Figure A.27. Example directory listing showing the two BINARY data files 1103B150, 1105B150, and 1111B075.

A.9.4 Analyzed Respiratory Data Files

When analyzed respiratory data is stored on a disk then a listing similar to the one shown in Figure A.28 is made. Be sure to ignore the "R" prefix when entering file names. The "R" is added to the file name automatically. Figure A.28 indicates that there are four files of analyzed respiratory data on the selected disk. The file names are 1109S200, 1111S200, 1204S150, and 1205S075.

```
------------------------------------------------------------------
 The following files may contain valid analyzed respiratory data.
 Ignore the leading R when entering the file names.
------------------------------------------------------------------

B9826:            Directory type= LIF level 0
created                   block size=256
 Storage order
...file name...    # blks    # bytes  last chng

R1109S200             102      25984 10-Feb-87
R1111S200             106      26992 10-Feb-87
R1204S150              60      15120 10-Feb-87
R1205S075              70      17696 10-Feb-87

FILES shown=4 allocated=28 unallocated=84
```

Figure A.28. Example directory listing showing four analyzed respiratory data files.

A.9.5 Exit to MAIN MENU

This menu selection is used to return to the main analysis menu without searching any disks.

A.10 Exit to System

This command is used to exit the main menu program. Control of the computer is returned to the HP Pascal operating system's command interpreter. To re-enter the menu at this time press "U", to execute the last USER program, or execute the program "HARDO:MENU".

CONTENTS - APPENDIX B

USING THE HP PASCAL OPERATING SYSTEM

APPENDIX B

USING THE HP PASCAL OPERATING SYSTEM

The breath-by-breath data collection and analysis programs are written in the Hewlett-Packard Pascal programming language. It is advisable that users know how to use the HP Pascal operating system. This appendix is not intended to replace the Pascal manuals. It is intended to be an introduction to HP Pascal.

B.1 General Information

B.1.1 Intended Audience

This appendix is written with the assumption that the user is not afraid of the computer and is willing to invest time and effort in familiarization with the computer system. Any user that desires to modify the breath-by-breath programs or study the algorithms should read this appendix.

B.1.2 Equipment Needed

It is assumed that the computer system used is the HP9826 located in the Bioengineering laboratory. At this time, the computer system contains an HP9826 computer with an internal five inch disk drive and 1.3 Mbytes of RAM. Interfaced on the HPIB (Hewlett Packard Interface Bus, also known as the IEEE-488) is an HP9895A eight inch floppy disk

drive, an HP9134A hard disk drive, and an HP2673A thermal printer. An HP General Purpose Input/Output (GPIO) interface is used to control the Data Acquisition Module (DAM). A Decwriter II dot matrix printer is connected using an RS-232C interface. Accessing the devices is discussed in Sections B.1.5, B.3, and B.5.

B.1.3 User Entries in Examples

All examples in this appendix that represent user input will have the following format. Prompts from the computer will be in upper case and a typical user response will be underlined. Replace the underlined example with the appropriate response.

B.1.4 File Names

Each file must be assigned a name. File names used by HP Pascal are different than those used by HP BASIC. All characters in a file name must be in capital letters. Pascal requires each name to have three parts. The three parts are the volume ID, name, and extension. For example, the file name "HARD1:ANALYSIS.TEXT" has volume ID "HARD1:", name "ANALYSIS", and extension ".TEXT". The extension .TEXT is the default file type created by the editor. The compiler generates a default .CODE extension. Another useful extension is .ASC, which indicates ASCII file types. ASCII files can be used to transfer data between Pascal and BASIC. ASCII files can be created by either the editor or by programs.

B.1.4.1 Work File

HP Pascal defines a default file that can be used for program

development. The file name is WORK.TEXT and it is stored on the current system volume. If a work file is used then the editor and compiler do not request file names. To rename or delete the work file, refer to Sections B.3.5 or B.3.6.

B.1.4.2 Wild Card Characters in File Names

When copying files and generating listings of disk directories, wild card characters may be used in the file name. Rather than explain the use of wild cards, examples using wild cards are presented in several of the examples in this appendix. The HP Pascal 3.0 Users Guide provides a complete description on the use of wild card characters in file names.

Description	Device number	Volume ID
Internal 5" drive	#3:	varies
External 8" drive	#7:	varies
Hard drive, platter 0	#11:	HARD0:
Hard drive, platter 1	#12:	HARD1:
Hard drive, platter 2	#13:	HARD2:
Hard drive, platter 3	#14:	HARD3:
RAM drive	#50:	RAM:
Thermal printer	#6:	PRINTER:
Decwriter printer	---	---
CRT	CONSOLE:	---
Keyboard	SYSTERM:	---
Plotter	---	---
Multiprogrammer	---	---

Table B.1. Summery of hardware devices by device number and ID.

B.1.5 Device Numbers and Volume IDs

Most of the hardware devices have an assigned device number. Table B.1 lists a summery of the devices that are available to the computer system. The device numbers are defined by the operating

system at boot time and can not be changed by the user. All of the disk drives also have an assigned volume ID. Volume ID's are useful because they can be changed by the user.

B.1.6 Default and System Volumes

Pascal defines two default volume settings. The default volume ID, also known as the default prefix, is the volume where all files are assumed to be if no volume ID is given. The system volume is the volume where the Pascal operating system stores system pointers and work files. After booting Pascal, both the default volume and system volume are set to HARD0:. Users may change the default settings of both volumes by using the WHAT command from the command interpreter. To minimize confusion, the system volume should be the same as the default volume.

B.2 Command Interpreter

The main level of operation in the Pascal system is the command interpreter. The command interpreter is a menu used to call other features such as the compiler and user programs. Located on the top of the screen, the prompt for the command interpreter is: ????

COMMAND: ASM DBG MEMV NEW PERM STREAM USER WHAT ?

A user can always return to the command interpreter by pressing the CLEAR IO/STOP key several times.

B.2.1 Executing (Running) Programs

To execute a program press the "X" key, for execute, from the

command interpreter. Next, enter the file name and press ENTER. In the following example, the user is executing the Main Menu program of the breath-by-breath respiratory measurement system.

EXECUTE WHAT FILE?: HARD0:MENU

The preceding example used a volume ID of HARD0: and a program name of MENU for a file name. Section B.1.4 describes the necessary format for file names.

B.3 Filer

The filer is a system utility that is loaded from the command interpreter by pressing the "F" key. The filer is used for disk directory maintenance such as copying files and making backups. The filer can also be used to interact with all of the volumes listed in Table B.1. This section provides a short description of several of the filer's features.

B.3.1 Directory Listings

From the filer press the "L" key for list directory. Enter either the device number or volume ID of the disk that you wish to view. For example,

LIST WHAT DIRECTORY? HARD3:

would list the directory of the volume labeled HARD3:. The same listing is displayed if device number #13: is entered instead of HARD3:. This can be verified by examining Table B-1.

Some examples using wild cards for directory listings are:

LIST WHAT DIRECTORY? HARD1:=.TEXT

creates a listing of all of the text files on volume HARD1:.

LIST WHAT DIRECTORY? HARD1:AN=.=

creates a list of all files on the volume HARD1: that start with the characters "AN".

B.3.2 Listing a Directory on the Printer

Adding the qualifier ",PRINTER" to a directory listing command will redirect the directory listing to the printer. For example:

LIST WHAT DIRECTORY? HARD1: ,PRINTER:

causes the directory listing to be sent to the system printer. The output can be sent to a disk file by replacing PRINTER: with a valid file name. Wild cards may be used in the file name specification.

B.3.3 Copying Files

To copy files press the "F" key for File copy. Enter the source file name then the destination file name. For example:

```
COPY WHICH FILE? DISPLAY.TEXT
TO WHICH FILE? #7:$
```

copies the program DISPLAY.TEXT from the default volume to the eight inch floppy disk. The dollar sign for a destination name tells the filer to use the same destination name as the source name. The dollar sign is not required if the entire destination name is entered. If a

different destination is desired then enter the new name in place of the dollar sign.

Wild cards can also be used to identify a group of source files that need to be copied. For example:

```
COPY WHICH FILE? HARD1:=.CODE
TO WHICH FILE? HARD2:$
```

copies all .CODE files from the volume HARD1: to HARD2:. In this example the dollar sign is necessary for the destination name.

B.3.4 Copying an Entire Disk

There are two ways to copy all of the files from one disk to another. The first method requires that both drives have the same storage requirements. The eight inch drive and the individual platters of the hard drive are equivalent in storage size. For example:

```
COPY WHICH FILE? HARD1:
TO WHICH FILE? #7:
DO YOU WISH TO DESTROY VOLUME #7? Y
```

copies the entire content of hard disk platter one to the eight inch floppy. All data on the eight inch disk will be destroyed by the new files. Note that the volume label of the eight inch disk will be changed to HARD1:. An error indicating this potential problem will be displayed on the screen. This method can be used to duplicate five inch disks in the following manner.

```
COPY WHICH FILE? #3:
TO WHICH FILE? #3:
INSERT THE DESTINATION DISK AND PRESS ENTER
DO YOU WISH TO DESTROY VOLUME #7 Y
```

The user will be required to swap the source and destination disks several times while the copy is being made. Be sure to place a write protect label on the source disk.

The second method of copying files does a file by file copy. The previous contents of the destination disk are not destroyed by the copy. This method can be used to copy between drives of different storage capacities.

```
COPY WHICH FILE? HARD1:=.=
TO WHICH FILE? #7:$
```

This technique copies all of the files from the source drive. The volume ID of the destination disk is not changed.

B.3.5 Renaming (Change) Files

To rename a file press the "C" key for file Change while in the system filer. For example:

```
WHAT FILE? HARD1:WORK.TEXT
CHANGE TO? MOUSE.TEXT
```

would rename the file WORK.TEXT on volume HARD1: to MOUSE.TEXT.

The file change function may also be used to change the volume ID of a disk. To change a volume label just enter the old and new volume IDs without file names. Be cautious with this option because the contents of the volume will be destroyed.

B.3.6 Deleting (Remove) Files

To delete files from a volume press the "R" key while in the filer. For example:

REMOVE WHICH FILE? WORK.TEXT

would be entered to delete the work file from the default volume. If wild cards are entered in the file name specification then all of the file names that match the description are listed. After the listing is made the user has the option of aborting the file deletion.

B.3.7 Crunching Disk Space

HP operating systems require that individual files must be stored as one contiguous block. As files are copied and deleted from a volume the disk will contain many small, unused blocks. The files may all be packed together by pressing the "K" key for crunch while in the filer. The amount of unused disk space and the largest contiguous space can be found in the summery of a directory listing.

B.4 Formatting Disks

All disks must be formatted before they can be used for data and program storage. Execute the program HARD0:MEDIAINIT to format a disk. Remember that the device identifier for the five inch disk is #3: and the eight inch disk is #7:. Disks can also be formatted using the HP BASIC command INITIALIZE.

B.5 Using the Decwriter (RS-232) Printer

The Pascal operating system permits only one printer to be defined in the system. The thermal printer is chosen as the default printer because it can be used for graphic screen dumps. Using the Decwriter II under program control is not difficult, however it is a nuisance. Refer to the program listings for ANALYSIS.TEXT, and DISPLAY.TEXT for examples that use the Decwriter II printer. A user can print .TEXT or .ASC files to the Decwriter by executing the program HARD1:LIST_FILE.

B.6 Using the Line Printer in the HP Laboratory

Because the Decwriter printer is slow a method of listing programs using the high speed line printer is presented. The procedure has several steps but is not difficult. The steps are:

1. Format a 5" disk. Place the disk in the 5" drive.
2. Load the program to be listed into the EDITOR.
3. Exit the editor using the Exit command.
4. Using the Write option, store the file on the 5" disk as an ASCII file. Use device number three and an .ASC file extension. For example, #3:ANALYSIS.ASC, stores the file on the 5" disk as an ASCII file with name ANALYSIS.
5. After the file is saved on the 5" disk, remove the disk and go to the HP computing laboratory.
6. Boot one of the HP9836 computers in the HP laboratory. Select NONE OF THE ABOVE for the class assignment.
7. Place the disk with the file to be printed in the right hand drive of the HP9836.
8. Obtain a directory listing of the disk using BASIC's catalog command. For example:

```
CATALOG ":INTERNAL"
```

9. Copy the file to the print spooler on the network. Use the file name that was displayed in step 8. There should be an "A" and some underline characters appended to the name. For example:

```
COPY "ANALYSISA_:INTERNAL" TO "SPOOL:ANALYSIS/REMOTE"
```

10. Remove the disk when the listing begins printing.

CONTENTS - APPENDIX C

DOCUMENTATION FOR THE PROGRAM MENU.TEXT

APPENDIX C

DOCUMENTATION FOR THE PROGRAM MENU.TEXT

This appendix contains documentation for the main menu program of the respiratory analysis programs. The main menu is used to execute other programs in the respiratory measurement system such as the calibration or data collection programs.

HP Pascal does not include a method of chaining or overlaying programs so this program uses a procedure that modifies the keyboard buffer's contents. The Pascal operating system is tricked into thinking the user entered the characters that execute the next program. For example, to execute the program ANALYSIS, which is stored on the volume HARD1: the keyboard driver procedure sends the contents of Figure C.1 to the keyboard's buffer. The maximum string length that may be sent to the keyboard buffer is 40 characters. The characters are sent to the buffer using the procedure Send_to_kbd.

```
XHARD1:ANALYSIS<cr>
|\   /\    /\ /
| \ /  \  /  V
|  V    \/    ------ Carriage return character
|  |     V           for the ENTER key.
|  |     |
|  |     ------------- Program's name.
|  |
|  ------------------- Volume ID.
|
---------------------- X for eXecute program.
```

Figure C.1. Characters sent to the keyboard buffer to execute the program HARD1:ANALYSIS.

<u>Global Variables - Program: MENU</u>

Choice	INTEGER variable equal to the menu choice entered by the user. The entry is checked for invalid entries.
Max	INTEGER variable equal to the maximum number of menu choices available to the user. Currently equal to seven.
Num_string	SHORT_STRING variable used for entry of numeric data by the user. HP Pascal crashes if the user presses a non-numeric key when Pascal is expecting a numeric entry. The procedure Value, described in the Utilities appendix, converts a string into an equivalent REAL value.
Valid_choice	BOOLEAN variable used for error checking the user's entry.

CONTENTS - APPENDIX D

DOCUMENTATION FOR THE PROGRAM CAP3.TEXT

APPENDIX D

DOCUMENTATION FOR THE PROGRAM CAP3.TEXT

This appendix contains documentation for version number three of the calibration program, CAP3. The calibration program is used to calculate calibration factors for use by the analysis program. Calibration factors are used to convert the binary data collected by the data acquisition module (DAM) into recognizable physical values such as temperature and fractional gas concentrations. The method used to control the DAM has not been modified since the previous version of this calibration program. For a description on how to control the DAM consult Riblett. This documentation is organized with all global variables and constants listed first, followed by procedures and functions in the same order as they appear in the listing.

The contents of the calibration file has been expanded since the previous version to include four, third order polynomial coefficients. The four curves represent the flow calibration factor as a function of the average binary reading calculated for a breath. There is one curve for inspiration, expiration, temperature and viscosity corrected inspiration, and temperature and viscosity corrected expiration. A third order fit adiquately describes the nonlinearity of the flow module. The constant flow calibration factors used by previous ver-

sions of this calibration program are still stored in the calibration file to allow compatiblity between versions.

Many of the variable and procedure descriptions that appear in this appendix are copied in whole or in part from Masters (3) and Riblett (2).

<u>Global Constants - Program: CAP3</u>

Flow_degree_of_fit	The degree of fit used for calibrating the flow module. Currently a third order fit is used.
Flow_num_pts_for_fit	The number of data points that will be fit for the flow calibration polynomial. Currently eight points are used for the fit.
Max_degree_of_fit	The maximum degree of fit that will be required by the curve fitting procedure. Currently set equal to Flow_degree_of_fit.
Max_num_pts_for_fit	The maximum number of points that will be required by the curve fitting procedure. Currently set equal to Temp_num_pts_for_fit.
Temp_degree_of_fit	The degree of fit used for temperature calibration. Currently a second order fit is used.
Temp_file_name	The file name used for the storage of the default temperature calibration coefficients.
Temp_num_pts_for_fit	The number of points that will be used for temperature calibration. Currently 20 points are used. The number of points should be an integer multiple of four because foure water baths are used for calibration.
Time_delay	The default GMS time delay in milliseconds that will be stored in the calibration file. Not used by this program because gas concentrations are assumed to be constant during GMS sampling. Currently set equal to 380 milliseconds.

<u>Global Variables - Program: CAP3</u>

Bin_zero_flow	INTEGER variable equal to the average binary value read from the flow channel for zero flow.

Calibration_stored
BOOLEAN variable. Set True after the calibration file has been stored to disk. Set False after calibrating the thermocouple, GMS, or flow module.

Cal_file_name — SHORT_STRING variable containing the file name used to load a calibration file from the disk.

CO2_cal — REAL value used to convert binary data collected from channel A of the DAM to fractional concentration values.

CO2_dc_offset — INTEGER value equal to the average binary value read from the CO_2 channel for 0% CO_2.

Corr_e_flow_cal — REAL variable containing the factor necessary to convert expiratory data collected from channel C of the DAM to values having flow units of liters per second.

Corr_flag — BOOLEAN variable. True if temperature and viscosity corrections are to be made to the flow calculations. False otherwise. Normally True.

Corr_incr_vol — REAL variable equal to the temperature and viscosity corrected incremental gas volume for the current time slice.

Corr_i_flow_cal — REAL variable containing the factor necessary to convert inspiratory data collected form channel C of the DAM to values having flow units of liters per second.

Date — SHORT_STRING variable equal to the date when the calibration file was made. Entered by the user.

Exit_program — BOOLEAN variable used in the menu selection routine. True if the program should return to the main menu program. Normally False.

Expr_c_poly — GLNARRAY variable array containing the four coefficients of the third order polynomial used to calculate the temperature and viscosity corrected flow calibration factor for expiration. The flow calibration factor is the value necessary to convert expiratory data collected from channel C of the DAM to values having flow units of liters per second. The [1] index represents the zero order term and the [4] index contains the third order term. The coefficients are loaded from the calibration file.

Expr_poly	GLNARRAY variable array containing the four coefficients of the third order polynomial used to calculate the non-corrected flow calibration factor for expiration. Refere to Expr_c_poly for a description of the subscripts of this array.
F	TEXT file descriptor required by Pascal when reading or writing to .ASC and .TEXT files.
Flow_calibrated	BOOLEAN variable. Normally False. Set True after the flow module is calibrated.
GMS_calibrated	BOOLEAN variable. Normally False. Set True after the O_2 and CO_2 channels have been calibrated.
GPINIT[7077890]	INTEGER variable pointing to a specific memory location. This memory location is part of the hardware interface of the DAM.
I	INTEGER variable used for loop counting and array indexing.
Insp_c_poly	GLNARRAY variable array containing the four coefficients of the third order polynomial used to calculate the temperature and viscosity corrected flow calibration factor for inspiration. Refere to Expr_c_poly for a description of the array subscripts.
Insp_poly	GLNARRAY variable array containing the four coefficients of the third order polynomial used to calculate the non-corrected flow calibration factor for inspiration. Refere to Expr_c_poly for a description of the array subscripts.
Line1	POINTER into the array of binary data collected from channel A of the DAM. 16 bit integers are used instead of 32 bit integers because only 12 bits are produced by the DAM. Currently the maximum number of points is 8000. This is maximum number of samples that will be required by any of the calibration procedures.
Line2	POINTER into the array of binary data collected from channel B of the DAM.
Line3	POINTER into the array of binary data collected from channel C of the DAM.
Line4	POINTER into the array of binary data collected from channel D of the DAM.

Listing	TEXT descriptor. Required to use the system printer.
Menu_choice	INTEGER variable equal to the user's selection from the calibration menu. Range checking is done to ensure that only valid entries are precessed.
MSI_device	SHORT_STRING variable set equal to the volume ID of the mass storage device that contains a desired disk file.
Non_c_e_flow_cal	REAL variable containing the factor necessary to convert expiratory data collected from the DAM to values having flow units of liters per second. Temperature and viscosity corrections are not included in the factor. This variable is no longer used but is retained in the program for compatability with earlier versions.
Non_c_incr_vol	REAL variable equal to the incremental volume of gas for the current time slice. Temperature and viscosity corrections are not included in this constant. This variable is no longer used but is retained in the program for compatability with earlier versions.
Non_c_i_flow_cal	REAL variable containing the factor necessary to convert inspiratory data collected from the DAM to values having flow units of liters per second. Temperature and viscosity corrections are not included in the factor. This variable is no longer used but is retained in the program for compatability with earlier versions.
Num_string	SHORT_STRING variable used for numeric inputs. HP Pascal crashes if a character is entered during a numeric read. Refer to the procedure Value, described in the utilities section, which converts a string to a real value.
O2_cal	REAL variable used to convert binary data collected from channel B of the DAM to fractional concentration units.
O2_dc_offset	INTEGER variable equal to the average binary value read from the O_2 channel of the DAM for 13.0% O_2.
O1	REAL variable equal to the actual O_2 concentration read from the mass spectrometer for 13% O_2.
Pb	REAL variable equal to the barometric pressure in

torr.

Q	SHORT_STRING variable used for all yes/no questions. The procedure Ask_y_or_n is used for entering and error checking this value.
Rel_humid	REAL variable equal to the relative humidity in fractional form.
Room_temp	REAL variable equal to the room temperature in degrees C. Entered by the user in degrees F.
S	INTEGER variable equal to the DAM sampling frequency in Hz.
Ta	REAL variable containing the second order temperature coefficient for converting DAM temperature data to degrees C.
Tb	REAL variable containing the first order temperature coefficient for converting DAM temperature data.
Tc	REAL variable containing the zero order temperature coefficient for converting DAM temperature data.
Temp_calibrated	BOOLEAN variable. True if the thermocouple has been calibrated. False indicates that the default calibration coefficients have been loaded.

Procedure: Hold_up

This procedure pauses program execution until the operator is ready to proceed. To continue program execution the operator must press the 'ENTER' key.

Variables passed to the procedure

None.

Variable returned by the procedure

None.

Variables used by the procedure

None.

Procedure: Curve_fit

This procedure performs an N^{th} order polynomial least squares curve fit to a set of input vectors. A fit greater than 6th order should not be attempted by this procedure. The matrix inversion technique uses LU decomposition and the corresponding procedures (Ludcmp and Lubksb) come directly from the text <u>Numerical Recipes The Art of Scientific Computing</u>, Press, Flannery, Teukolsky, Vetterling, p. 31-38, 683-684 and are documented there. The least squares fit used generates a matrix of coefficients, $\underline{P}$ by the following technique.

$$\underline{P} = (\underline{A}^T\underline{A})^{-1}\underline{A}^T\underline{Y}$$

$$\underline{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \\ \\ y_m \end{bmatrix} \qquad \underline{A} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ & & & & \\ & & & & \\ 1 & x_m & x_m^2 & \cdots & x_m^n \end{bmatrix}$$

$$\underline{P} = \begin{bmatrix} \text{Zero order coefficient} \\ \text{First order coefficient} \\ \\ \\ N^{th} \text{ order coefficient} \end{bmatrix}$$

n=The degree of fit desired.
m=The number of points to be used for the fit.

<u>Variables passed to the procedure</u>

Degree INTEGER variable equal to the degree of fit to use.

Num_pts	INTEGER variable equal to the number of (X,Y) points to use for the fit.
X_vector	GLNARRAY array variable containing the X coordinates.
Y_vector	GLNARRAY array variable containing the Y coordinates.

<u>Variables returned by the procedure</u>

Poly	GLNRESULT array variable containing the coefficients of the calculated polynomial. Poly[1] contains the zero order coefficient.

<u>Variables used by the procedure</u>

A	GLNSQUARE A square matrix used for temporary storage. Refer to program documentation for implementation of least-squares curve fitting method.
A_tran	GLNPBYNP Transpose of A matrix.
B	GLNRESULT Matrix product: B=A_tran*Y_vector.
C	GLNSQUARE Inverse of the matrix $(\underline{A}^T\underline{A})$.
Col	INTEGER variable use for array indexing.
D	INTEGER variable used by the LU decomposition and back-substitution procedures.
Deg	INTEGER variable used for array indexing.
I	INTEGER variable used for array indexing.
Indx	INTEGER variable used for array indexing.
J	INTEGER variable used for array indexing.
N	INTEGER variable equal to the degree of fit plus one.
Row	INTEGER variable used for array indexing.
Sum	REAL variable used for intermediate calculations during matrix multiplication.

Procedure: Clkset

The procedure Clkset sets the 8253 timer chip in the DAM with the desired sampling frequency. A maximum frequency of 350 Hz is recommended. 50 Hz is currently the normal operating frequency. This procedure was designed by Riblett and has not been modified.

<u>Variables passed to this procedure</u>

S — INTEGER variable containing the desired sampling frequency in Hz.

<u>Variables returned by this procedure</u>

None.

<u>Variables used by this procedure</u>

Fl — INTEGER variable used to set the least significant byte of clock 1 of the DAM's 8253 timer chip to the proper sampling rate.

Fm — INTEGER variable used to set the most significant byte of clock 1 of the DAM's 8253 timer chip to the proper sampling rate.

X — INTEGER variable used to calculate Fl and Fm.

Procedure: Data_collect

This procedure controls the sampling requirements of the DAM. Currently only four channels are sampled. Data_collect calls the assembly language procedures HIT_HI and BIT_TST to test the status of the DAM. These two assembly language procedures are not listed in this documentation. See Riblett for a description of the two procedures.

Variables passed to this procedure

S — INTEGER variable equal to the sampling rate in Hz.

Sam — INTEGER variable equal to the number of samples that are to be taken.

Variable returned by this procedure (Global variables)

Line1 — POINTER into the CO_2 binary data array.

Line2 — POINTER into the O_2 binary data array.

Line3 — POINTER into the flow binary data array.

Line4 — POINTER into the temperature binary data array.

Variables used by this procedure

Del — INTEGER variable used in the delay loop that allows the sample/hold amplifiers time to track the input signals before the hold command is given.

I — INTEGER variable used for loop counting and array indexing.

R4 — INTEGER variable used to read from the DAM's status register.

R6 — INTEGER variable used to write to the DAM's control register.

Procedure: Gascal

This procedure is used to calibrate the Perkin-Elmer gas mass spectrometer for the CO_2 and O_2 channels.

Variables passed to this procedure

S	INTEGER variable equal to the sampling rate in Hz.

Variables returned by this procedure

CO2_cal	REAL value used to convert binary data collected from channel A to fractional concentration values.
CO2_dc_offset	INTEGER variable equal to the average binary value read from channel A for 0% CO_2.
O1	REAL variable equal to the actual O_2 concentration read from the mass spectrometer for 13% O_2.
O2_cal	REAL variable used to convert binary data collected from channel B to fractional concentration values.
O2_dc_offset	INTEGER variable equal to the average binary value read from the O_2 channel of the DAM for 13% O_2.
S	INTEGER variable equal to the sampling rate in Hz.

Variables used by this procedure

Avg_CO2_cal_gas	INTEGER variable equal to the average binary value read for 7% CO_2.
Avg_CO2_room_air	INTEGER variable equal to the average binary value read for 0% CO_2.
Avg_O2_cal_gas	INTEGER variable equal to the average binary value read for 17% O_2.
Avg_O2_room_air	INTEGER variable equal to the average binary value read for 21% O_2.
Ch	REAL variable equal to the actual fractional concentration read from the mass spectrometer for 7% CO_2.
Cl	REAL variable equal to the actual fractional concen-

tration read from the mass spectrometer for 0% CO_2.

I	INTEGER variable used for loop counting and array indexing.
Oh	REAL variable equal to the actual fractional concentration read from the mass spectrometer for 21% O_2.
Sam	INTEGER variable set equal to the number of samples that are to be taken.

Procedure: Flowcal

The procedure Flowcal calculates the calibration factors for the flow signal. The calibration factors returned to the main program are the coefficients of a third order fit between average binary values and the conversion factor from binary readings to flow in liters per second. The binary value for zero flow is first determined then two sets of flow calibration coefficients are determined. The first two calibrations do not use temperature or gas viscosity corrections in their determinations. The second set uses temperature and viscosity corrections in their calculations.

<u>Variables passed to this procedure</u>

Corr_flag	BOOLEAN variable. True if temperature and viscosity corrections are to be used in the determination of flow calibration factors.
Pb	REAL variable equal to the barometric pressure in torr.
Rel_humid	REAL variable equal to the relative humidity.
Room_temp	REAL variable equal to the room temperature in ^{o}C.
S	INTEGER variable equal to the sampling rate in Hz.
Ta, Tb, Tc	REAL variables containing the temperature calibration coefficients.

<u>Variables returned by this procedure</u>

Bin_zero_flow	INTEGER
Corr_e_flow_cal	REAL variable equal to the T&V corrected expiratory flow calibration factor.

Corr_i_flow_cal REAL variable equal to the T&V corrected inspiratory flow calibration factor.

Expr_c_poly GINARRAY array variables containing the third order coefficients for the T&V corrected expiratory flow calibration factor.

Expr_poly GINARRAY array variables containing the third order coefficients for the non-T&V corrected expiratory flow calibration factor.

Insp_c_poly GINARRAY array variables containing the third order coefficients for the T&V corrected inspiratory flow calibration factor.

Insp_poly GINARRAY array variables containing the third order coefficients for the non-T&V corrected inspiratory flow calibration factor.

Non_c_e_flow_cal
REAL variable equal to the non-T&V corrected expiratory flow calibration factor.

Non_c_i_flow_cal
REAL variable equal to the non-T&V corrected inspiratory flow calibration factor.

Variables used by this procedure

A INTEGER variable used as an index into the flow signal array during flow signal integration.

Array_pointer INTEGER variable used as an index into the water vapor pressure array Vap.

Bin_delta_p INTEGER variable equal to the binary differential pressure developed by the pneumotach. This is equal to the sampled value minus zero flow.

Corr_aire REAL variable used in the expiratory integration loop to sum up the binary volume of air expired by the calibration syringe or pump using the flow calculated under STPD conditions.

Corr_airi REAL variable used in the expiratory integration loop to sum up the binary volume of air inspired by the calibration syringe or pump using the flow calculated under STPD conditions.

Corr_avg_vol_expr
REAL variable equal to the average binary volume

expired by the calibration syringe or pump under STPD conditions.

Corr_avg_vol_insp
REAL variable equal to the average binary volume inspired by the calibration pump or syringe under STPD conditions.

Corr_pump_vol — REAL variable equal to the volume in liters of the calibration pump or syringe converted to STPD conditions.

Corr_tot_vol_expr
REAL variable equal to the total STPD volume expired by all pump or syringe cycles.

Corr_tot_vol_insp
REAL variable equal to the total STPD volume inspired by all pump or syringe cycles.

Expr_cal_factor — GLNARRAY array containing the non-temperature and viscosity corrected expiratory flow calibration factors calculated for each of the eight flow rates.

Expr_c_cal_factor
GLNARRAY array containing the temperature and viscosity (STPD) expiratory flow calibration factors calculated for each of the eight flow rates.

Expr_c_flow — GLNARRAY array containing the average STPD expiratory stroke volumes for each of the eight flow rates.

Expr_flow — GLNARRAY array containing the average non-temperature and viscosity corrected expiratory stroke volumes for each of the eight flow rates.

FH2O — REAL variable equal to the fractional concentration of water vapor in the room air.

File_name — SHORT_STRING variable equal to the file name used to store the flow points and calibration curves.

F_ASCII — TEXT file descriptor. Required by Pascal for all ASCII and text file I/O.

Gas_CO2 — REAL variable equal to the actual fractional concentration of CO_2 in the gas used for flow calibration.

Gas_N2 — REAL variable equal to the actual fractional concentration of N_2 in the gas used for flow calibration.

Gas_O2 — REAL variable equal to the actual fractional concentration of O_2 in the gas used for flow calibration.

I — INTEGER variable used for loop counting and array indexing.

Insp_cal_factor — GLNARRAY array containing the non-temperature and viscosity corrected inspiratory flow calibration factors calculated for each of the eight flow rates.

Insp_c_cal_factor — GLNARRAY array containing the temperature and viscosity (STPD) inspiratory flow calibration factors calculated for each of the eight flow rates.

Insp_c_flow — GLNARRAY array containing the average STPD inspiratory stroke volumes for each of the eight flow rates.

Insp_flow — GLNARRAY array containing the average non-temperature and viscosity corrected inspiratory stroke volumes for each of the eight flow rates.

J — INTEGER variable used for loop counting and array indexing.

Non_c_aire — REAL variable used in the expiratory integration loop to sum up the binary volume of air inspired by the calibration pump or syringe in ATPS conditions.

Non_c_airi — REAL variable used in the inspiratory integration loop to sum up the binary volume of air inspired by the calibration pump or syringe in ATPS conditions.

Non_c_avg_vol_expr — REAL variable equal to the average binary volume expired by the calibration pump or syringe in ATPS.

Non_c_avg_vol_insp — REAL variable equal to the average binary volume expired by the calibration pump or syringe in ATPS.

Non_c_pump_vol — REAL variable equal to the volume in liters of the calibration pump or syringe under ATPS conditions.

Non_c_tot_vol_expr — REAL variable equal to the total volume expired by all pump or syringe cycles in ATPS

Non_c_tot_vol_insp — REAL variable equal to the total volume inspired by all pump or syringe cycles in ATPS.

No_breaths	INTEGER variable equal to the number of breaths counted during flow calibration.
Num_expr_points	INTEGER variable equal to the total number of expiratory calibration points for this cycle.
Num_for_good	INTEGER variable equal to the minimum number of either inspiratory or expiratory points that must be present in order for the cycle to be determined good.
Num_insp_points	INTEGER variable equal to the total number of inspiratory calibration points for this cycle.
Sam	INTEGER variable equal to the number of samples that are to be taken.
Still_expire	BOOLEAN variable. True when the flow is expiratory.
Still_inspire	BOOLEAN variable. True when the flow is inspiratory.
T	REAL variable equal to the reciprocal of the sampling frequency S.
Temperature	REAL variable equal to the gas temperature in ^{o}C.
Tot_expr_points	INTEGER variable equal to the total number of calibration points for all pump cycles determined to be expiratory.
Tot_insp_points	INTEGER variable equal to the total number of calibration points for all pump cycles determined to be inspiratory.
Tot_zero	INTEGER variable equal used in the calculation of zero flow.
Vap	REAL ARRAY containing water vapor pressure values from 20.0 to 44.9 ^{o}C in 0.1 ^{o}C increments.
Vap_file	FILE OF REAL values. Data values are read into the vapor pressure array Vap.
Volume_name	SHORT_STRING variable equal to the volume ID of the mass storage device that will be used for storing the flow points and curves.

Procedure: Valid_pointer

This procedure is used to determine if the index into the saturated water vapor array is within the limits of the array. This procedure was originally designed with the conception that the fractional concentration of water vapor in the flow gas would be determined from the instantaneous temperature of the flow gas. The Valid_pointer procedure protects against temperatures computed from bad temperature data points from the DAM. In this case the computed temperature would point to a water vapor pressure value which is beyond the bounds of the vapor pressure array causing the program to crash. This procedure protects against this possibility by checking to make certain the water vapor index is within bounds.

<u>Variables passed to this procedure</u>

Array_pointer INTEGER variable. On entry contains the initial array index.

<u>Variables returned by this procedure</u>

Array_pointer INTEGER variable. On exit contains an array index that is within the range of the vapor pressure array.

<u>Variables used by this procedure</u>

None.

Procedure: Calc_incr_vol

This procedure calculates the incremental volume of the flow sample from the measured binary differential pressure. The incremental volume is then used for numerical integration of the flow signal. The non T&V corrected flow is assumed to be a function of the pressure differential.

<u>Variables passed to this procedure</u>

Bin_delta_p	INTEGER variable equal to the measured differential pressure minus the calculated binary zero flow.
FH2O	REAL variable equal to the fractional concentration of water vapor.
T	REAL variable equal to the sampling period.
Temperature	REAL variable equal to the temperature in degrees C of the sampled gas.

<u>Variables returned by this procedure</u>

Corr_incr_vol	REAL value equal to the T&V corrected binary volume.
Non_c_incr_vol	REAL value equal to the non T&V corrected binary volume.

<u>Variables used by this procedure</u>

ITPnS_to_STPD	REAL variable that converts the sampled volume to STPD conditions.
Rel_viscosity	REAL variable equal to the viscosity of the gas relative to the viscosity of room air under STPD conditions.

Function: Rel_viscos

This function returns a real value equal to the viscosity of the flow gas relative to the viscosity of room air under STPD conditions.

Variables passed to this procedure

FH2O	REAL value equal to the fractional concentration of water vapor in room air.
Temperature	REAL value equal to the temperature in degrees C.

Variables returned by this procedure

None.

Variables used by this function

FCO2	REAL variable equal to the fractional concentration of O_2 in wet room air.
FN2	REAL variable equal to the fractional concentration of N_2 in wet room air.
FO2	REAL variable equal to the fractional concentration of O_2 in wet room air.
Vis_CO2	REAL variable equal to the viscosity of CO_2 calculated as a function of gas temperature.
Vis_H2O	REAL variable equal to the viscosity of water calculated as a function of gas temperature.
Vis_N2	REAL variable equal to the viscosity of N_2 calculated as a function of gas temperature.
Vis_O2	REAL variable equal to the viscosity of O_2 calculated as a function of gas temperature.
Vis_STPD	REAL variable equal to the viscosity of room air under STPD conditions.

Procedure: Tempcal

This procedure calculates the calibration factors for the temperature transducer. The temperature calibration factors are the coefficients of a second order polynomial. The polynomial is calculated from a least-squares-fit of twenty sampled water baths. Tempcal allows for errors in the thermometer to be accounted for. The constant Thermometer_cal_Y_int must be set equal to the y intercept of the thermometer's calibration curve. Set Thermometer_cal_slope equal to the slope of the calibration curve. Currently these parameters are set equal to an intercept of zero and a slope of one.

<u>Variables passed to this procedure</u>

S	INTEGER equal to the sampling frequency in Hz.

<u>Variables returned by this procedure</u>

Ta	REAL value equal to the second order coefficient of the temperature calibration curve.
Tb	REAL value equal to the first order coefficient.
Tc	REAL value equal to the zero order coefficient.

<u>Variables used by this procedure</u>

F	TEXT file descriptor required for Pascal to write text files to the disk.
I	INTEGER variable used for loop counting and array indexing.
Loop	INTEGER variable used for loop counting and array indexing. Counts the bath number being used at the current time.

Sam	INTEGER variable equal to the number of samples that are to be taken for each data point.
Sum	INTEGER variable used for calculating the average of the binary values read for the current bath.
Temp_binary	GLNARRAY variable array containing the binary readings for each of the 20 bath measurements.
Temp_deg_C	GLNARRAY variable array containing the true temperature in degrees C of the 20 bath measurements.
Temperature_poly	GLNARRAY variable array containing the coefficients of the second order temperature polynomial. These coefficients are transferred to the variables Ta, Tb, and Tc.

Procedure: Load_tempcal

This procedure loads the default temperature calibration coefficients from a disk file. The name of the file is stored in the global constant Temp_file_name, which currently equals the string "HARD0:TEMP_CRV.ASC".

<u>Variables passed to this procedure</u>

None.

<u>Variables returned by this procedure</u>

Ta	REAL value equal to the second order coefficient of the temperature calibration curve.
Tb	REAL value equal to the first order coefficient.
Tc	REAL value equal to the zero order coefficient.

<u>Variables used by this procedure</u>

F	TEXT file descriptor required for Pascal to read text files from the disk.

CONTENTS - APPENDIX E

APPENDIX E

DOCUMENTATION FOR THE PROGRAM DAP3.TEXT

This appendix documents version three of the data acquisition program. The data acquisition program is used to collect respiratory data from a subject. Collected data is stored in disk files for processing by the program ANALYSIS. Real time analysis of the respiratory signals is not currently available. This documentation is organized with all global variables and constants listed first, followed by procedures and functions in the order they appear in the program listing. The file UTILITIES.TEXT, which contains eight utility procedures and functions, merges into this program when compiled. Refer to the utilities appendix for descriptions of the procedures and functions included in that file.

Many of the variable and procedure descriptions that appear in this appendix are copied in whole or in part from Masters (3) and Riblett (2).

<u>Global Constants - Program: DAP3</u>

Max	The maximum number of samples per channel that may be collected. Currently equal to 60000. At a sampling rate of 50 Hz, up to 30 minutes of data may be collected.

Global Variables - Program: DAP3

Abort	BOOLEAN variable. True if the user wishes to exit the program without collecting any data. Normally False.
CO2max	INTEGER variable equal to the maximum binary reading from the CO_2 channel.
CO2min	INTEGER variable equal to the minimum binary reading from the CO_2 channel.
Fmax	INTEGER variable equal to the maximum binary reading from the flow channel.
Fmin	INTEGER variable equal to the minimum binary reading from the flow channel.
Line1	POINTER into the array that stores the binary data from channel A.
Line2	POINTER into the array that stores the binary data from channel B.
Line3	POINTER into the array that stores the binary data from channel C.
Line4	POINTER into the array that stores the binary data from channel D.
O2max	INTEGER variable equal to the maximum binary reading from the O_2 channel.
O2min	INTEGER variable equal to the minimum binary reading from the O_2 channel.
Tmax	INTEGER variable equal to the maximum binary reading from the temperature channel.
Tmin	INTEGER variable equal to the minimum binary reading from the temperature channel.

Procedure: Hold_up

This procedure pauses program execution until the operator is ready to proceed. To continue program execution the operator must press the "ENTER" key.

<u>Variables passed to the procedure</u>

None.

<u>Variable returned by the procedure</u>

None.

<u>Variables used by the procedure</u>

None.

Procedure: Clkset

The procedure Clkset sets the 8253 timer chip in the DAM to the desired sampling frequency. A maximum frequency of 350 Hz is recommended. 50 Hz is the normal operating frequency. This procedure was designed by Riblett. The only addition is the abort techniqe for exiting the program.

Variables passed to this procedure

S	INTEGER variable containing the desired sampling frequency in Hz.

Variables returned by this procedure

Abort	BOOLEAN variable. Set True if the user enters a negative value for the sampling frequency. The calling procedure will exit the program without collecting data if Abort is True. Normally False.

Variables used by this procedure

Fl	INTEGER variable used to set the least significant byte of clock 1 of the DAM's 8253 timer chip to the proper sampling rate.
Fm	INTEGER variable used to set the most significant byte of clock 1 of the DAM's 8253 timer chip to the proper sampling rate.
X	INTEGER variable used to calculate Fl and Fm.

Procedure: Maxmin

This procedure determines the maximum and minimum binary values collected from the CO_2, O_2, flow and temperature channels. The maximum and minimums are displayed on the CRT so the operator has an idea of the input signal range. The number of points out of range for each channel are also calculated and displayed on the CRT.

Variables passed to this procedure

Sam	INTEGER value equal to the number of samples stored in memory.

Variables returned by this procedure

CO2max	INTEGER variable equal to the maximum binary reading from the CO_2 channel.
CO2min	INTEGER variable equal to the minimum binary reading from the CO_2 channel.
Fmax	INTEGER variable equal to the maximum binary reading from the flow channel.
Fmin	INTEGER variable equal to the minimum binary reading from the flow channel.
O2max	INTEGER variable equal to the maximum binary reading from the O_2 channel.
O2min	INTEGER variable equal to the minimum binary reading from the O_2 channel.
Tmax	INTEGER variable equal to the maximum binary reading from the temperature channel.
Tmin	INTEGER variable equal to the minimum binary reading from the temperature channel.

Variables used by this procedure

I	INTEGER variable used for loop counting and array indexing.

Num_CO2_equal_0	INTEGER variable equal to the number of samples from the CO_2 channel equal to zero.
Num_CO2_equal_4095	INTEGER variable equal to the number of samples from the CO_2 channel equal to 4095.
Num_F_equal_0	INTEGER variable equal to the number of samples from the flow channel equal to zero.
Num_F_equal_4095	INTEGER variable equal to the number of samples from the flow channel equal to 4095.
Num_O2_equal_0	INTEGER variable equal to the number of samples from the O_2 channel equal to zero.
Num_O2_equal_4095	INTEGER variable equal to the number of samples from the O2 channel equal to 4095.
Num_T_equal_0	INTEGER variable equal to the number of samples from the temperature channel equal to zero.
Num_T_equal_4095	INTEGER variable equal to the number of samples from the temperature channel equal to 4095.

Procedure: Data_storage

This procedure stores the four channels of binary data to disk files. The user must enter the root of the four file names and choose a volume ID. The CO_2 file name is the root preceded with a "C". The O_2 file name is formed by prededing the root with an "O". The flow file name is preceded by an "F" and the temperature file name is preceded by a "T".

<u>Variables passed to this procedure</u>

CO2max	INTEGER variable equal to the maximum binary reading from the CO_2 channel.
CO2min	INTEGER variable equal to the minimum binary reading from the CO_2 channel.
Fmax	INTEGER variable equal to the maximum binary reading from the flow channel.
Fmin	INTEGER variable equal to the minimum binary reading from the flow channel.
Line1	POINTER into the array for channel A.
Line2	POINTER into the array for channel B.
Line3	POINTER into the array for channel C.
Line4	POINTER into the array for channel D.
O2max	INTEGER variable equal to the maximum binary reading from the O_2 channel.
O2min	INTEGER variable equal to the minimum binary reading from the O_2 channel.
Tmax	INTEGER variable equal to the maximum binary reading from the temperature channel.
Tmin	INTEGER variable equal to the minimum binary reading from the temperature channel.

Variables returned by this procedure

None.

Variables used by this procedure

F_2	FILE OF INT_16BIT. File descriptor for the 16-bit integer arrays that will be stored disk files.
Good_save	BOOLEAN variable. True if the data was saved to the disk without error. False if an I/O error occurred during the saving process. If an I/O error occurs then the operator is given another chance to save the binary data.
I	INTEGER variable used for loop counting and array indexing.
MSI_device	SHORT_STRING variable equal to the volume ID of the mass storage device that will contain the disk file.
Q	SHORT_STRING variable used for answers to yes/no questions.

Procedure: Data_collect

This procedure controls the sampling requirements of the DAM. Currently, only four channels are sampled. Data_collect calls the assembly language procedures HIT_HI and BIT_TST to test the status of the DAM. These two assembly languages procedures are not listed in this documentation. Refer to Riblett for descriptions of the two assembly language procedures.

Variables passed to this procedure

S	INTEGER variable equal to the sampling rate in Hz.
Sam	INTEGER variable equal to the number of samples that are to be taken.

Variable returned by this procedure

Line1	POINTER into the CO_2 binary data array.
Line2	POINTER into the O_2 binary data array.
Line3	POINTER into the flow binary data array.
Line4	POINTER into the temperature binary data array.

Variables used by this procedure

Delay	INTEGER variable used in the delay loop that allows the sample/hold amplifiers time to track the input signals before the hold command is given.
I	INTEGER variable used loop counting and array indexing.
R4	INTEGER variable used to read from the DAM's status register.
R6	INTEGER variable used to write to the DAM's control register.

CONTENTS - APPENDIX F

DOCUMENTATION FOR THE PROGRAM ANALYSIS3.TEXT

APPENDIX F

DOCUMENTATION FOR THE PROGRAM ANALYSIS3.TEXT

The analysis program is used to calculate the respiratory parameters from the binary data collected using the data acquisition program, DAP3, and the calibration program, CAP3. An operator can analyze an entire set of data or a window. Windowing the data permits the respiratory variables to be calculated during a steady state of exercise. Analyzed data can be stored in a disk file for additional filtering and plotting by the DISPLAY program.

This documentation is organized with all global variables and constants listed first, followed by the procedures and functions. The procedures and functions are listed in the order that they occur in the program listing. All local variables are defined alphabetically. Many of the variable, procedure, and function descriptions that appear in this appendix are copied in whole or in part from Riblett (2) and Masters (3).

Global constants - ANALYSIS3

Degree_of_fit	The polynomial degree of fit for the inspiratory and expiratory calibration curves.
First_element_vap	The index to use for the first record in the saturated vapor pressure array.
Last_element_vap	The index to use for the last record in the saturated vapor pressure array.
Max	The maximum number of samples per DAM channel that may be analyzed.
Max_resp_index	The number of respiratory parameters that are stored in the array Resp_var_array.
Max_temp	The maximum temperature which may be found in the vapor pressure array Vap.
Min_temp	The minimum temperature which may be found in the vapor pressure array Vap.
Print_file_name	The file name used for temporary storage of text files to be printed by the RS-232 printer driver procedure Copy_2_printer.
Vap_file_name	The saturated vapor pressure file name.

The following fourteen constants are indexes into the simulated two dimensional respiratory-variable array Resp_var_array. The descriptions pertain to what is stored in the indexed cell.

Breath_good_index	Equals 1 if this was a good breath, 0 otherwise.
CO_prod_index	CO_2 produced in liters.
Expr_min_vent_index	The expiratory minute ventilation.
FRC_CO2_prod_index	The alveolar CO_2 produced.
FRC_O2_cons_index	The alveolar O_2 consumed.
FRC_resp_quot_index	The alveolar respiratory quotient.
Insp_min_vent_index	The inspiratory minute ventilation.
O2_cons_index	The volume of O_2 consumed this breath.
Resp_freq_index	The current respiratory frequency.

Resp_quotient_index The respiratory quotient.

Time_of_brth_index The time at the start of the breath.

V_tid_insp_index The inspiratory tidal volume.

V_tid_expr_index The expiratory tidal volume.

Volume_of_lung_index
The end-expiration lung volume.

Global type declarations - ANALYSIS3

Int_16bit This is a subrange of INTEGER that requires only sixteen bits of storage for each variable. A normal HP Pascal integer requires 32 bits. The short integer is used for the binary data returned by the DAM because the DAM returns 12-bit data.

Binary_array Dynamic array type for the storage of DAM binary data. Line1, ..., Line4 will be defined as pointers into arrays of this type.

Short_string A useful string length for miscellaneous string operations such as file names and numeric reads.

Long_string A useful string length for miscellaneous string operations such as printed comments.

Flow_cal_poly_type Defines an array size for storing the inspiratory and expiratory flow calibration polynomial coefficients.

Kbd_ctrl_str An array of characters required for program control of the operating system's keyboard driver. Used only the the procedure Send_2_kbd.

Global_variables - ANALYSIS3

A | INTEGER variable used as an index into the flow and temperature signal arrays during signal integration. See also Z.

A	INTEGER variable used as an index into the flow and temperature signal arrays during signal integration. See also Z.
Age	REAL variable equal to the age of the subject. Used by the FRC calculations for the default initial lung volume.
Air_expr	REAL variable equal to the amount of air expired during the current breath in liters.
Air_insp	REAL variable equal to the amount of air inspired for the current breath in liters.
Ave_expr_binary	REAL variable indicating the average binary value measured from the DAM during the current breath. Used as the argument of the expiratory calibration factor function Expr_cal_factor.
Ave_insp_binary	REAL variable indicating the average binary value measured from the DAM during the current breath. Used as the argument of the inspiratory calibration factor function Insp_cal_factor.
Avg_air_expr	REAL variable equal to the average amount of gas expired per breath in liters.
Avg_air_insp	REAL variable equal to the average amount of gas inspired per breath in liters.
Avg_CO2_expr	REAL variable equal to the average amount of CO_2 expired per breath in liters.
Avg_CO2_insp	REAL variable equal to the average amount of CO_2 inspired per breath in liters.
Avg_CO2_prod	REAL value equal to the average CO_2 produced per breath in liters.
Avg_O2_cons	REAL variable equal to the average O_2 consumed per breath in liters.
Avg_O2_expr	REAL variable equal to the average amount of O_2 expired per breath in liters.
Avg_O2_insp	REAL variable equal to the average amount of O_2 inspired per breath in liters.
Avg_time_delay	REAL value equal to the average of the valid time delays in milliseconds as determined by the

breath-by-breath time delay procedure Bbb_time_delay.

Begtime_colct	REAL value entered by the user. This time is added to all time calculations. Allows the first data point to occur at a time other than time equal to zero. Default time is zero seconds.
Binary_file_name	The file name used as a template for loading the four channels of collected DAM data. CO_2 files have a prefix of "C", O_2 files have an "O" prefix, flow files have an "F" prefix, and temperature data has a "T" prefix. The file name is also used as a template for storing analyzed respiratory data. "R" and "A" prefixes are used for the two analyzed data files.
Bin_zero_flow	INTEGER variable equal to the average binary value read from the flow channel for zero flow. Loaded from the calibration file.
Body_temp	REAL variable equal to the end exercise body temperature in degrees C.
Breath_count	INTEGER variable equal to the total number of breaths analyzed.
BTPS_to_STPD	REAL variable used as a multiplying constant to convert from BTPS to STPD conditions. See also STPD_to_BTPS.
B_by_b_output	BOOLEAN variable. True if the breath-by-breath data is to be printed. False if only the averages of the respiratory variables are to be printed.
Calibration_available	BOOLEAN variable. True if a calibration file has been loaded. False otherwise.
Cal_file_name	SHORT_STRING variable containing. The file name used to load the calibration file. The volume ID is not included.
CO2_cal	REAL value used to convert the binary data collected from channel A of the DAM to fractional concentration values. Loaded form the calibration file.
CO2_dc_offset	INTEGER value equal to the average binary value read from the CO_2 channel for 0% CO_2. Loaded from the calibration file.

CO2_expr	REAL variable equal to the amount of CO_2 expired during this breath in liters.
CO2_insp	REAL variable equal to the amount of CO_2 inspired during this breath in liters.
CO2_max	INTEGER variable equal to the maximum reading from the CO_2 channel. See also O2_max, F_max, and T_max.
CO2_min	INTEGER variable equal to the minimum reading from the CO_2 channel.
CO2_prod	REAL variable equal to the amount of CO_2 produced for the current breath in liters.
Comment	LONG_STRING variable equal to the comment printed on the result summary of analysis.
Corr_e_flowcal	REAL variable containing the factor necessary to convert expiratory data collected from channel C of the DAM to values having flow units of liters per second. This variable is loaded from the calibration file but is no longer used.
Corr_flag	BOOLEAN variable. True if temperature and viscosity corrections are to be made to the flow calculations. False otherwise.
Corr_i_flowcal	REAL variable for the inspiratory portion of the flow calibration. See Corr_e_flowcal for a further description. This variable is loaded from the calibration file but is no longer used.
Data_available	BOOLEAN. True if binary data has been loaded into memory. False otherwise.
Date	SHORT_STRING variable equal to the date the calibration file was made. Analysis assumes that the calibration date is the same as the date when data was collected. Loaded from the calibration file.
End_point	INTEGER variable equal to the index in the binary data arrays that analysis is to end. See also start_point.
Exit_program	BOOLEAN variable used in the menu selection routine. True if the program should return to the main menu program. Normally False.
Expr_begin	INTEGER variable used as a pointer to where

	expiration first begins within the flow signal. This pointer is used along with the last expiration point to determine the time of expiration for the current breath. See also Insp_count.
Expr_corr_cal_poly	REAL ARRAY [1..4] variable containing the four coefficients of the third order polynomial used to calculate the temperature and viscosity corrected flow calibration factor for expiration. The flow calibration factor is the factor necessary to convert expiratory data collected from channel C of the DAM to values having flow units of liters per second. The [1] index contains the zero order term and the [4] index contains the third order term. The coefficients are obtained from the calibration file. See also Insp_corr_cal_poly, Expr_non_corr_cal_poly, and Insp_non_corr_cal_poly.
Expr_flowcal	REAL variable containing a non-temperature and viscosity corrected flow calibration factor similar to Corr_e_flow_cal. This factor is loaded from the calibration file but is no longer used.
Expr_non_corr_cal_poly	REAL ARRAY [1..4] variable containing the four coefficients of the polynomial used to calculate the non-temperature corrected flow calibration factor. See Expr_corr_cal_poly for further information.
Expr_time	REAL variable equal to the time elapsed during the current expiration in seconds.
F	TEXT file declaration used for writing to the print file that will be printed using the procedure Send_2_printer.
FCO2	REAL variable equal to the fractional concentration of CO_2 at the current time. Used to calculate the volume of CO_2 present in the current incremental time slice and by the Calc_sig_vol procedure for viscosity corrections. Calculated in the Calc_sig_vol procedure.
FH2O_sat	REAL ARRAY containing water vapor pressure values from 20.0 to 44.9 ^{o}C divided by the barometric pressure in torr. See also Vap.
File_ASCII	TEXT file descriptor used for writing the useful calibration and analysis information to the disk

when the analyzed respiratory data are to be stored in a file.

File_name_ASCII	SHORT_STRING variable equal to the file name of the file that the useful calibration and analysis information is to be stored in.
File_REAL	FILE of REAL values used for writing the analyzed respiratory data to a disk file.
Final_index	INTEGER variable equal to the last expiratory index of the analysis. Used to calculate the total time of respiration.
Final_insp	INTEGER variable equal to the first index of the last inspiration.
Flow_error	REAL variable equal to the inspiratory vs expiratory alveolar flow error. In percent.
FN2	REAL variable equal to the fractional concentration of N_2 at the current time. Used to calculate the volume of N_2 present in the current incremental time slice and by the Calc_sig_vol procedure for viscosity corrections. Calculated in the Calc_sig_vol procedure.
FO2	REAL variable equal to the fractional concentration of O_2 at the current time. Used to calculate the volume of O_2 present in the current incremental time slice and by the Calc_sig_val procedure for viscosity corrections. Calculated in the Calc_sig_vol procedure.
Frac_CO2_2	REAL variable used for FRC alveolar calculations. Contains the fractional concentration of CO_2 for the previous breath.
Frac_H2O_body	REAL variable equal to the saturated fractional concentration of water at body temperature.
Frac_N2_2	REAL variable used for FRC alveolar calculations. Contains the fractional concentration of N_2 for the previous breath.
Frac_O2_2	REAL variable used for FRC alveolar calculations. Contains the fractional concentration of O_2 for the previous breath.
FRC	REAL variable equal to the functional residual volume of the subject. Approximated from the subject's age, weight, and height. Refer to

Rick's work for development of FRC corrections.

FRC_CO2_prod	REAL variable equal to the amount of alveolar CO_2 produced in liters during the current breath. See also FRC_O2_cons and FRC_RQ.
FRC_flag	BOOLEAN variable. True if FRC alveolar calculations are to be made. False otherwise. Permanently set True in the procedure Set_defaults.
FRC_O2_cons	REAL variable equal to the amount of alveolar O_2 consumed in liters during the current breath. See also FRC_CO2_prod.
FRC_RQ	REAL variable equal to the respiratory quotient for the current breath. See also FRC_CO2_prod.
F_max	INTEGER variable equal to the maximum reading from the flow channel. See also CO2_max, O2_max, and T_max.
F_min	INTEGER variable equal to the minimum reading from the flow channel. See also CO2_min, O2_min, and T_min.
Good_expr_count	INTEGER variable equal the number of good expirations found during the analysis.
Good_insp_count	INTEGER variable equal to a count of the number of good inspirations found during the analysis.
Height	REAL variable equal to the subject's height in cm. Entered in units of inches from either the procedure Change_defaults or loaded from the file FRC_INFO.TEXT.
I	INTEGER variable used for loop counting and indexing.
Incr_vol	REAL variable returned from procedure Calc_sig_val equal the incremental volume of air for the current time slice.
Init_index	INTEGER index into the flow signal array indicating where analysis of the respiratory data begins. Init_index is used with Final_index to determine total respiratory time in seconds.
Insp_corr_cal_poly	REAL ARRAY [1..4] variable containing the four coefficients of the polynomial used to calculate the temperature corrected flow calibration factor

in the procedure Insp_cal_factor. See Expr_corr_cal_poly for further information.

Insp_count — INTEGER variable used as an index to where inspiration first begins within the flow signal. This index is used along with the end of inspiration point to determine the time of inspiration for the current breath. See also Expr_begin.

Insp_flowcal — REAL variable containing the factor necessary to convert inspiratory data collected from the DAM to values having flow units of liters per second. Temperature and viscosity corrections are included in the factor. The value is loaded from the calibration file but is no longer used by this program because it was replaced by a polynomial curve.

Insp_non_corr_cal_poly — REAL ARRAY [0..3] variable containing the four coefficients of the polynomial used to calculate the non-temperature corrected flow calibration factor in the procedure Insp_cal_factor. See Expr_corr_cal_poly for further information.

Insp_time — REAL variable equal to the time of inspiration in seconds.

Line1 — PT1 pointer into the dynamic array of binary data collected from channel A of the DAM. 16 bit integers are used instead of 32 bit integers because only 12 bits are produced by the DAM. Currently the maximum number of points is contained in the constant MAX which equals 60000.

Line2 — PT2 pointer into the dynamic array of binary data collected from channel B of the DAM.

Line3 — PT3 pointer into the dynamic array of binary data collected from channel C of the DAM.

Line4 — PT4 pointer into the dynamic array of binary data collected from channel D of the DAM.

Listing — TEXT file descriptor required to use the system printer as an output device.

Lung_vol — REAL variable equal to the current volume of the lung.

Max_index — INTEGER pointer into the CO_2 data string where the

maximum CO_2 fractional concentraion is observed in the current breath. Used to determine the GMS time delay value and FRC alveolar parameters.

Menu_choice	INTEGER value equal to the menu selection from the analysis menu. Range checking is done to ensure only valid entries are processed.
Min_vol_e	REAL variable equal to the expiratory minute volume of the subject in liters per minute.
Min_vol_i	REAL variable equal to the inspiratory minute volume of the subject in liters per minute.
N2store	REAL variable equal to the volume of N2 stored in the lung. Part of the FRC calculations. Refer to Rick's work for development.
N2_expr	REAL variable equal to the amount of expired N_2 for the current breath in liters.
N2_insp	REAL variable equal to the amount of inspired N_2 for the current breath in liters.
Net_lung_chng	REAL variable equal to the volume of the lung's change for the current breath. Part of the FRC calculations.
Non_c_e_flowcal	REAL variable containing the factor necessary to convert expiratory data collected from the DAM to values having flow units of liters per second. Temperature and viscosity corrections are not included in the factor. The value is loaded from the calibration file but is no longer used because it was replaced by a polynomial curve.
Non_c_i_flowcal	REAL variable containing the non-corrected inspiratory flow calibration factor. The value is loaded from the calibration file but is no longer used.
No_points	INTEGER variable equal to the total number of data points stored in memory.
Num_breaths	INTEGER variable equal to the number of breaths analyzed.
Num_string	SHORT_STRING variable used for numeric reads. HP Pascal crashes if a non-numeric character is entered during numeric reads. Refer to the procedure Value, described in the Utilities section, which converts a string to a number of

type REAL.

O2_cal	REAL variable used to convert binary data collected from channel B of the DAM to fractional concentration units. Loaded from the calibration file.
O2_cons	REAL variable equal to the amount of O_2 consumed during the current breath in liters.
O2_dc_offset	INTEGER variable equal to the average binary value read from the O_2 channel for 13.0% O_2. Loaded from the calibration file.
O2_expr	REAL variable equal to the amount of expired O_2 during the current breath in liters.
O2_insp	REAL variable equal to the amount of inspired O2 for the current breath in liters.
O2_max	INTEGER variable equal to the maximum reading from the O_2 channel. See also CO2_max, F_max, and T_max.
O2_min	INTEGER variable equal to the minimum reading from the O_2 channel.
O1	REAL variable containing the actual O_2 concentration read from the mass spectrometer for 13.0% O_2. Loaded from the calibration file.
Pb	REAL variable containing the barometric pressure in torr. Loaded from the calibration file or entered in inches of Hg.
PH2O_body	REAL variable equal to the vapor pressure of water at body temperature in torr.
Q	SHORT_STRING variable used for all yes/no questions. See also the procedure Ask_y_or_n.
R	REAL variable equal to the respiratory quotient.
Rel_humid	REAL variable equal to the relative humidity in fractional form. Loaded from the calibration file or entered in percent form.
Resp_freq	REAL variable equal to the respiratory frequency of the subject in breaths per minute.
Resp_var	ARRAY of REAL variables equal to the respiratory parameters that will be stored in the analyzed

	respiratory data file. The single dimensional array is stored to the disk after each breath is calculated.
Room_fH2O	REAL variable containing the fractional concentration of water in room air.
Room_temp	REAL value equal to the room temperature in degrees C. Loaded from the calibration file or entered in ^{o}F.
S	INTEGER variable equal to the DAM sampling frequency in Hz. See also T.
Start_point	INTEGER variable containing the index into the DAM data arrays that analysis is to begin. See also End_point.
Still_calc	BOOLEAN variable. True until all DAM data points have been analyzed.
Stop_analysis	BOOLEAN variable. True if the user pressed a key during the analysis indicating that they wish to abort the calculations. Normally False.
Store_anal_data	BOOLEAN variable. True if the analyzed breath-by-breath data should be stored in a disk file.
STPD_to_BTPS	REAL variable used as a multiplying constant to convert from STPD to BTPS conditions. See also BTPS_to_STPD.
Subject_name	SHORT_STRING variable containing the subject's name. The name is used for loading the default settings for the subject's age, height, and weight.
T	REAL variable equal to the reciprocal of the sampling frequency. See also S.
Ta	REAL variable containing the second order coefficient for converting DAM temperature data to degrees C. Loaded from the calibration file.
Tb	REAL variable containing the first order coefficient for converting DAM temperature data to degrees C.
Tc	REAL variable containing the zero order coefficient for converting DAM temperature data to degrees C.

Time_delay	INTEGER variable representing the gas mass spectrometer time delay in milliseconds. Calculated in the the procedure Bbb_time_delay.
Time_delay_cnt	INTEGER variable containing the number of valid time delays computed by the breath-by-breath time delay procedure Bbb_time_delay. This count is used to calculate the average time delay.
Time_delay_flag	INTEGER flag which equals zero if fixed time delays are requested, one if variable time delays are requested, and two if variable time delays are requested but the time delay calculated for the current breath was invalid.
Time_delay_sum	REAL variable equal to the sum of all valid time delays.
Tot_CO2_expr	REAL variable equal to the total volume of expired CO_2 in liters.
Tot_CO2_insp	REAL variable equal to the total volume of inspired CO_2 in liters.
Tot_CO2_prod	REAL variable equal to the total volume of CO_2 produced in liters.
Tot_FRC_CO2_prod	REAL variable equal to the total volume alveolar CO_2 produced in liters.
Tot_FRC_O2_cons	REAL variable equal to the total volume alveolar O_2 consumed in liters.
Tot_N2store	REAL varialbe equal to the total volume of N_2 stored in the lung. Part of the FRC calculations.
Tot_O2_cons	REAL variable equal to the total volume of O_2 consumed in liters.
Tot_O2_expr	REAL variable equal to the total volume of O_2 expired in liters.
Tot_O2_insp	REAL variable equal to the total volume of O_2 inspired in liters.
Tot_time	REAL variable equal to the total time in seconds of inspiration and expiration for the current breath.
Tot_time_expr	REAL variable equal to the total time of expiration in seconds.

Tot_time_insp	REAL variable equal to the total time of inspiration in seconds.
Tot_time_resp	REAL variable equal to the total time of respiration in seconds.
Tot_vol_expr	REAL variable equal to the total volume of expired gas in liters.
Tot_vol_insp	REAL variable equal to the total volume of inspired gas in liters.
T_max	INTEGER variable equal to the maximum reading from the temperature channel. See also CO2_max, F_max, and O2_max.
T_min	INTEGER variable equal to the minimum reading from the temperature channel.
V1	REAL variable equal to the initial FRC calculation. See also FRC.
Vap	REAL ARRAY containing water vapor pressure values from 20.0 to 44.9 ^{o}C in increments of 0.1 ^{o}C. The values are loaded from a disk file. See also FH2O_sat.
Vap_file	FILE descriptor of REAL variables required to load the vapor pressure data into the array Vap.
V_dot_CO2	REAL variable equal to the average rate CO_2 is produced in liters per minute.
V_dot_FRC_CO2	REAL variable equal to the average rate alveolar CO_2 is produced in liters per minute.
V_dot_FRC_O2	REAL variable equal to the average rate alveolar O_2 is consumed in liters per minute.
V_dot_O2	REAL variable equal to the average rate O_2 is consumed in liters per minute.
Weight	REAL variable equal to the subject's weight in Kg. Loaded from the file FRC_INFO.TEXT or entered by the user.
Z	INTEGER variable used as an index into the CO_2 and O_2 signal arrays during signal integration. See also A.

Procedure: Copy_2_printer

This procedure copies a text file from disk to the Decwriter II RS-232 printer. This procedure is required because HP Pascal only allows one system printer to be defined. The thermal printer is defined as the system printer so it can be used for graphic screen dumps. An XON/XOFF handshake protocol is implemented because the printer's buffer overflows during continuous printing. Although the HP 9826 supports hardware handshaking, the DecWriter's interface does not.

Variables passed to this procedure

None.

Variables returned from this procedure

None.

Constants defined by this procedure

Prt_device	The hardware HPIB select code for the printer that will receive the outgoing text. The RS-232 interface has select code 9 while the thermal printer is 701.
XOFF	The standard ASCII character for Stop-transmit.
XON	The standard ASCII character for Start-transmit.

Variables used by this procedure

C	CHARACTER variable used to store characters received from the RS-232 serial interface.
Line	STRING[200] variable used for storing the text lines that will be sent to the serial printer.
T	TEXT file descriptor required for text file I/O.

Function: Insp_cal_factor

This function returns an inspiratory flow calibration factor that is a function of the average binary value measured for a given breath. The calibration factor is currently represented by a 3rd order polynomial calculated during calibration. The boolean operator Corr_flag determines whether the temperature corrected calibration factor curve or the non-corrected curve should be used.

Variables passed to this function

Ave_binary	REAL value equal to the average binary conversion for the current breath. This is the argument for the polynomial curve.
Corr_flag	BOOLEAN value. True if the T&V corrected calibration curve is to be used. False otherwise.

Variables returned by this function

None.

Variables used by this function

I	INTEGER variable used for loop control.
Temp	REAL variable used for intermediate calculations.

Function: Expr_cal_factor

This function returns an expiratory flow calibration factor that is a function of the average binary value measured for a given breath. The calibration factor is currently represented by a 3rd order polynomial calculated during calibration. The boolean operator Corr_flag determines whether the temperature corrected calibration factor curve or the non-corrected curve should be used.

Variables passed to this function

Ave_binary	REAL value equal to the average binary conversion for the current breath. This is the argument for the polynomial curve.
Corr_flag	BOOLEAN value. True if the T&V corrected calibration curve is to be used. False otherwise.

Variables returned by this function

None.

Variables used by this function

I	INTEGER variable used for loop control.
Temp	REAL variable used for intermediate calculations.

Procedure: Load_FRC_subject_info

This procedure loads the exercising subject's age in years, height in inches, and weight in pounds from a disk file. This information is required by the FRC calculations of a default initial lung volume. If the subject's name is not found then the three parameters will remain unchanged. Height and Weight are convert to metric units before the procedure exits.

Variables passed to this procedure

Subject_name	SHORT_STRING variable containing the name that will be searched for in the file of defaults.

Variables returned by this procedure

Age	REAL value equal to the subject's age in years.
Height	REAL value equal to the subject's height in cm.
Weight	REAL value equal to the subject's weight in Kg.

Variables used by this procedure

F_text	TEXT file descriptor required for working with text files.
Found_it	BOOLEAN variable. True if a name in the file matched the requested name. False otherwise.
Test_name	SHORT_STRING variable loaded from the text file. Used for matching the variable Subject_name.
Trash_line	LONG_STRING variable used to store the four or five leading lines of the file. The leading lines describe the text file's format.

Function: Valid_pointer

This INTEGER function determines if the vapor pressure array index is within the proper range. If the index is within range then this function returns the index's value. If the index is out of range then this function returns the closest index that is within range. This function is required because the vapor pressure array is only valid from 20° to 40°C and the room temperature often falls below 20°C during the winter budget cuts.

Variables passed to this function

Array_pointer — INTEGER variable equal to the vapor pressure array pointer that may be out of range.

Variables returned by this function

None.

Variables used by this function

None.

Procedure: Set_defaults

This procedure is used to set the default settings of the variables that can be altered by the user. The default settings permit a novice user to analyze breath-by-breath data without understanding of all of the terms.

<u>Variables passed to this procedure</u>

None.

<u>Variables returned by this procedure</u>

None.

<u>Global variables modified by this procedure</u>

Age	REAL variable equal to the subject's age in years.
Avg_time_delay	REAL variable equal to the average time delay.
Begtime_colct	REAL variable equal to the time when data collection started. Defaults to zero seconds.
Body_temp	REAL variable equal to the body temperature in ^{o}C.
B_by_b_output	BOOLEAN variable. Set False so the breath-by-breath data will not be printed for each breath.
Comment	LONG_STRING variable set equal to the null string.
Corr_flag	BOOLEAN variable set true so T&V corrections are made.
FRC_flag	BOOLEAN variable set true so FRC calculations are made.
Height	REAL variable equal to the subject's height in cm.
Pb	REAL variable equal to the barometric pressure in torr.
Rel_humid	REAL variable equal to the relative humidity.
Room_temp	REAL variable equal to the room temperature in ^{o}C.
S	INTEGER variable equal to the sampling frequency.

Time_delay	REAL variable equal to the default GMS time delay.
Time_delay_flag	INTEGER variable. Set equal to 1 so breath-by-breath time delays are calculated.
Time_delay_sum	REAL variables equal to the sum of all valid time delays. Defaults to the same value as Time_delay.
Weight	REAL variable equal to a default weight for the subject.

Procedure: Change_defaults

This procedure allows is used to change the default settings of the analysis variables. The procedure Set_defaults initializes all of the variables to practical values during the program initialization.

Variables passed to this procedure

None.

Variables returned by this procedure

None.

Variables used by this procedure

I	INTEGER variable used for loop couning and indexing.
Num_string	SHORT_STRING variable used for numeric inputs. Refer to Num_string description in the Global variables section of this appendix.

Global variables modified by this procedure

Age	REAL variable equal to the subject's age in years.
Avg_time_delay	REAL variable equal to the average time delay.
Begtime_colct	REAL variable equal to the time when data collection started. Defaults to zero seconds.
Body_temp	REAL variable equal to the body temperature in ^{o}C.
B_by_b_output	BOOLEAN variable. Set False so the breath-by-breath data will not be printed for each breath.
Comment	LONG_STRING variable set equal to the null string.
Corr_flag	BOOLEAN variable set true so T&V corrections are made.
FRC_flag	BOOLEAN variable set true so FRC calculations are made.
Height	REAL variable equal to the subject's height in cm.

Pb	REAL variable equal to the barometric pressure in torr.
Rel_humid	REAL variable equal to the relative humidity.
Room_temp	REAL variable equal to the room temperature in ^{o}C.
S	INTEGER variable equal to the sampling frequency.
Time_delay	REAL variable equal to the default GMS time delay.
Time_delay_flag	INTEGER variable. Set equal to 1 so breath-by-breath time delays are calculated.
Time_delay_sum	REAL variables equal to the sum of all valid time delays. Defaults to the same value as Time_delay.
Weight	REAL variable equal to a default weight for the subject.

Procedure: Hard_copy_head

This procedure prints a header on the top of the analysis printout. The text htat is to be printed is directed to a disk file that will be printed on the DecWriter using the procedure Copy_2_printer.

Variables passed to this procedure

None.

Variables returned by this procedure

None.

Variables used by this procedure

I	INTEGER variable used as a loop counter.
Print_file	TEXT file descriptor required by Pascal for all text file I/O. Used to create a print file for the procedure Copy_2_printer.

Procedure: Hard_output

This procedure prints the breath-by-breath parameters to a disk file for printing by the procedure Copy_2_printer. Hard_out is only called if breath-by-breath output has been selected in the modify parameters section of the analysis menu.

<u>Variables passed to this procedure</u>

None.

<u>Variables returned by this procedure</u>

None.

<u>Variables used by this procedure</u>

None.

Procedure: Means

This procedure prints the time averaged respiratory data. The information to be printed is first stored in a disk file then printed using the procedure Copy_2_printer.

Variables passed to this procedure

None.

Variables returned by this procedure

None.

Variables used by this procedure

F	TEXT file descriptor required by Pascal for all text file I/O. Used to create a print file for the procedure Copy_2_printer.
G	SHORT_STRING variable. If temperature corrections were made then G is set equal to the units "BTPS", otherwise G is set equal to a blank string.
H	SHORT_STRING variable. If temperature corrections were made then H is set equal to the units "STPD", otherwise H is set equal to a blank string.
I	INTEGER variable used as a loop counter.
Temporary	REAL variable used for intermediate calculations.

Procedure: Plot_data

This procedure contains all of the graphics routines required to plot a window of the binary data on the computer's CRT or plotter.

Variables passed to this procedure

Time_delay	REAL value equal to the GMS time delay in milliseconds. Required to time align the CO_2 and O_2 signals.

Variables returned by this procedure

None.

Variables used by this procedure

Aspect_ratio	REAL variable required to use the entire surface of the plotting device.
Again	BOOLEAN variable used for loop control.
Bot	REAL variable equal to the virtual coordinate of the bottom edge of the current graphic window.
Cmax	REAL variable equal to the maximum value of the CO_2 data array in the data block that is to be plotted.
Cmin	REAL variable equal to the maximum value of the CO_2 data array in the data block that is to be plotted.
Error_return	INTEGER variable used when initializing the HP Pascal graphic routines. Non-zero if a graphics error occurred during initialization.
Fmax	REAL variable equal to the maximum of the flow signal.
Fmin	REAL variable equal to the minimum of the flow signal.
GMS_time_delay_offset	INTEGER variable equal to the number of samples needed for time alignment of the O_2 and CO_2 signals.
I	INTEGER variable used for loop counting and

indexing.

Label_header	STRING[100] variable used for storage of text strings for the GTEXT plotting utility.
Left	REAL variable equal to the virtual coordinate of the left edge of the current graphic window.
Omax	REAL variable equal to the maximum of the O_2 signal.
Omin	REAL variable equal to the minimum of the CO2 signal.
Plot_device	INTEGER variable used to inform the HP Pascal graphics procedures which device to use for plotting. A value of 3 indicates the CRT while 705 indicates the plotter.
Right	REAL variable equal to the virtual coordinate of the right edge of the current graphic window.
Temp	REAL variable used for temporary work space.
Tmax	REAL variable equal to the maximum of the temperature signal.
Tmin	REAL variable equal to the minimum of the temperature signal.
Top	REAL variable equal to the virtual coordinate of the top edge of the current graphic window.
Width	INTEGER variable equal the number of points to be plotted. Used for calculating the variables Left and Right.

Procedure: Set_pen_speed

This procedure is used to set the plotter's pen speed to a known value. By default, the plotter runs at full blast, which is 30 cm/sec maximum. A typical maximum pen speed for good plots is 5 cm/sec. This procedure is only called from the procedure Plot_data.

This procedure is copied from page 85 of the HP Pascal 3.0 Graphics Procedures Manual.

Variables passed to the procedure

Speed — INTEGER variable equal to the desired maximum pen speed in cm/sec. A maximum value of 30 cm/sec is allowed by the current eight-pen vector plotter.

Variables returned by the procedure

None.

Procedure: Label_plot

This procedure draws and labels a set of axes within the current graphic window. The procedure is only called by the procedure Plot_data.

Variables passed to this procedure

Bot	REAL variable equal to the virtual coordinate of the bottom edge of the current graphic window.
Center_pt	INTEGER value equal to the Y intercept of the independent axis.
End_pt	INTEGER value equal to the index of the last data point to be plotted.
Label1	LONG_STRING variable equal to the dependent axis label.
Label2	LONG_STRING variable equal to the dependent axis units.
Label3	LONG_STRING variable equal to the independent axis label and units.
Left	REAL variable equal to the virtual coordinate of the left edge of the current graphics window.
Max	REAL value equal to the maximum dependent axis value.
Min	REAL value equal to the minimum dependent axis value.
Right	REAL variable equal to the virtual coordinate of the right edge of the current graphics window.
Start_pt	INTEGER value equal to the index of the first data point to be plotted.
Top	REAL variable equal to the virtual coordinate of the top edge of the current graphics window.

Variables returned by this procedure

None.

Variables used by this procedure

Blow_off	INTEGER variable used for temporary work space.
Char_height	REAL variable equal to the height that characters are to be drawn using the GTEXT graphics procedure.
Char_size	REAL variable equal to the percent of the screen size that each graphic character should be drawn. Used to form the variables Char_height and Char_width. For example, if the plotting surface is to be 40 characters wide then set Char_size to 1/40 or 0.025.
Char_width	REAL variable equal to the width that characters are to be drawn using the GTEXT graphics procedure.
I, J	INTEGER variables used for loop counters and indexing.
Num_string	SHORT_STRING variable used for numeric inputs. Refer to Num_string description in the Global variables section of this appendix.
Temp1, Temp2	REAL variables used for temporary work space.
Width	INTEGER variable equal to the difference between the indicies of the first point and the last point to plot.

Procedure: FRC_calc

Using the end fractional concentrations of the previous breath, this procedure calculates alveolar O_2 consumption and CO_2 production. This procedure was originally written by Rick Pieschl.

Variables passed to this procedure

None.

Variables returned by this procedure

None.

Variables used by this procedure

Chng_fCO2	REAL variable equal to the change in the fractional volume of CO_2 for the current breath.
Chng_lung_vol	REAL variable equal to the change in the lung volume for the current breath.
Chng_N2	REAL variable equal to the change in the fractional volume of N_2 for the current breath.
Chng_fO2	REAL variable equal to the change in the fractional volume of O_2 for the current breath.
Frac_CO2_1	REAL variable equal to the fractional concentration of CO_2 for the current breath.
Frac_N2_1	REAL variable equal to the fractional concentration of N_2 for the current breath.
Frac_O2_1	REAL variable equal to the fractional concentration of O_2 for the current breath.

Procedure: Bbb_time_delay

This procedure is used to calculate an appropriate time delay for the gas mass spectrometer (GMS). A time delay calculation is required because the GMS does not give an instantaneous reading of the gasses present at the tip of the sampling capillary. Approximately 380 milliseconds are required for samples to travel the length of the capillary and pass through the GMS. Consult Riblett (2) for an in-depth discussion of how the time delay is calculated.

Variables passed to this procedure

A	INTEGER variable equal to the current index into the respiratory binary data arrays Line1..Line4.

Variables returned by this procedure

Time_delay	REAL value equal to the calculated time delay in msec.

Variables used by this procedure

Adiff	REAL variable representing the absolute value of the difference between Asum and Bsum. By minimizing Adiff, the breath-by-breath time delay can be determined.
Asum	REAL variable containing the area above the CO_2 signal based upon the integration limits Beg_pt and Beg_intg.
Beg_intg	INTEGER variable used allong with End_intg for the numerical integration while attempting to make the volumes equal.
Beg_pt	INTEGER array pointer for the CO_2 signal which points to where integration above the CO_2 signal begins.
Best_index	INTEGER array pointer which points to that

	location in the CO_2 and O_2 signal arrays corresponding to the beginning of inspiration. By also knowing where inspiration begins, the breath-by-breath time delay can be determined.
Best_match	REAL variable containing the smallest difference in the area above and area below the fractional CO_2 signal.
Bomb_out	BOOLEAN variable used to flag the exit from the procedure. True when the best time delay is found.
Bsum	REAL variable containing the area below the CO_2 signal based on the integration limits Beg_intg and End_pt.
C	INTEGER variable set equal to the CO2_dc_offset.
CO2_max	INTEGER variable equal to the maximum CO_2 value for the current breath.
CO2_min	INTEGER variable equal to the minimum CO_2 value for the current breath.
End_intg	INTEGER variable used along with Beg_intg for the numerical integration while to attempting to make the volumes equal.
End_pt	INTEGER array pointer for the CO_2 signal which points to where integration of the CO_2 signal ends. Points to the same location as the pointer Min_index.
Found_the_max	BOOLEAN variable used for exiting loops.
Mid_index	INTEGER pointer into the CO_2 data array where 1/2 of the maximum CO_2 fraction is observed in the current breath.
Min_index	INTEGER pointer into the CO_2 data array where the minimum CO_2 fraction is observed in the current breath.
Wye	INTEGER variable used for loop counting and indexing.
Z	INTEGER variable used as an initial approximation for the index pointing to the maximum CO_2 concentration.

Procedure: Calc_sig_vol

This procedure calculates the incremental binary volume for the current time slice. The incremental volume is calculated then converted to STPD conditions if temperature and viscosity corrections are requested. The fractional concentrations of CO_2, N_2, and O_2 are calculated and used in the determination of the relative viscosity of the gas.

Variables passed to this procedure

A — INTEGER variable used to index into the flow and temperature data arrays.

Insp_or_expr — SHORT_STRING variable. If equal to "INSP" then the data is assumed to be inspiratory and room air is used for fractional concentrations of water vapor. Otherwise, the gas is assumed to be expiratory and saturated with water vapor.

Z — INTEGER variable used to index into the CO_2 and O_2 data arrays.

Variables returned by this procedure

FCO2 — REAL value equal to the instantaneous fractional concentration of CO_2.

FN2 — REAL value equal to the instantaneous fractional concentration of N_2.

FO2 — REAL value equal to the instantaneous fractional concentration of O_2.

Incr_vol — REAL value equal to the binary volume of the sampled gas for this increment of time.

Variables used by this procedure

Bin_flow — REAL variable used in an intermediate calculation of the temperature and viscosity corrected incremental volume.

FCO2_wet	REAL variable equal to the fractional concentration of CO_2 in saturated gas at the current temperature.
FH2O	REAL variable equal to the fractional concentration of water in air at the current temperature.
FN2_wet	REAL variable equal to the fractional concentration of N2 in saturated gas at the current temperature.
FO2_wet	REAL variable equal to the fractional concentration of O_2 in saturated gas at the current temperature.
Rel_viscosity	REAL variable equal to the relative viscosity of the sampled gas. Returned from the function Rel_viscos.
Temp	INTEGER variable equal to the binary temperature read from the DAM.
Temperature	REAL variable equal to the temperature in degrees C. Calculated from the second order polynomial using the coefficients Ta, Tb, and Tc.
Vis_CO2	REAL variable equal to the relative viscosity of CO_2 calculated as a linear function of temperature.
Vis_H2O	REAL variable equal to the relative viscosity of water calculated as a linear function of temperature.
Vis_N2	REAL variable equal to the relative viscosity of N_2 calculated as a linear function of temperature.
Vis_O2	REAL variable equal to the relative viscosity of O_2 calculated as a linear function of temperature.

Function: Inspiration

The inspiration function tests a flow sample to determine if the flow is inspiratory. The function returns True if one of the following two conditions are met.

1. The flow sample is less than the binary zero flow, or
2. one of the next five flow samples is less than or equal to zero flow.

This function has the effect of filtering bad ADC conversions.

Variables passed to this function

Pntr	INTEGER value equal to the current index into the flow signal array.

Variables returned by this function

None.

Variables used by this function

I	INTEGER variable used for loop control and indexing.
Insp	BOOLEAN variable used as temporary storage for the result of this function.
Diff	INTEGER variable equal to the difference between the measured binary flow from the DAM and binary zero flow.

Function: Expiration

The expiration function tests a flow sample to determine if the flow is expiratory. The function returns True if one of the following two conditions are met.

1. The flow sample is greater than the binary zero flow, or
2. one of the next five flow samples is greater than or equal to zero flow.

This function has the effect of filtering bad ADC conversions.

Variables passed to this function

Pntr	INTEGER value equal to the current index into the flow signal array.

Variables returned by this function

None.

Variables used by this function

I	INTEGER variable used for loop control and indexing.
Expr	BOOLEAN variable used as temporary storage for the result of this function.
Diff	INTEGER variable equal to the difference between the measured binary flow from the DAM and binary zero flow.

Procedure: Load_data

This procedure loads four channels of binary data from a disk file. The user must enter the file name that the four files are stored under, and the mass storage device that contains the files. The program then precedes the file name entered with the following characters to form file names for the four binary files.

'C' precedes the file name for CO_2 files.

'O' precedes the file name for O_2 files.

'F' precedes the file name for N_2 files.

'T' precedes the file name for temperature files.

The subject's name is also loaded. The subject's name is required to load default FRC parameters.

Variables passed to this procedure

None.

Variables returned by this procedure

None.

Variables used by this procedure

C_file_name	SHORT_STRING variable equal to the file name used for loading the CO_2 data file.
F	FILE of INT_16BIT. File descriptor for the 16-bit integer data files stored by the data acquisition program DAP3.
F_file_name	SHORT_STRING variable equal to the file name used for loading the flow data file.
Good_name	BOOLEAN variable used by the error trapping procedure. True indicates that all file I/O was

	accomplished without error. False indicates that an error occurred during file I/O.
I	INTEGER variable used for loop counting and indexing.
MSI_device	SHORT_STRING variable set equal to the volume ID of the mass storage device that contains the file.
O_file_name	SHORT_STRING variable equal to the file name used to load the O_2 data file.
T_file_name	SHORT_STRING variable equal to the file name used to load the temperatory data file.

Procedure: Load_cal

This procedure loads a calibration file from a disk file. The user must enter the file name and the mass storage device that contains the calibration information.

Variables passed to this procedure

None.

Variables returned by this procedure

None.

Variables used by this procedure

F	TEXT file descriptor. Required for working with .TEXT and .ASC files.
File_name	SHORT_STRING variable set equal to the file name of the data to be loaded.
Good_name	BOOLEAN variable used by the error trapping procedure. True indicates that all file I/O was accomplished without error. False indicates that an error occurred during file I/O.
I	INTEGER variable used for loop counting and indexing.
MSI_device	SHORT_STRING variable equal to the volume ID of the mass storage device that contains the file.

Procedure: Store_data

This procedure stores the analyzed respiratory data array, Resp_var, to a disk file. The file name defaults to the same string that the Load_data procedure used to load the binary data. The file name is preceded with an "R" for Respiratory data. Another file is created using an "A" instead of the "R". The second file contains useful calibration information about this run. The program DISPLAY reads the files stored by this procedure.

Variables passed to this procedure

Breath_count	INTEGER variable equal to the total number of breaths stored in the file.

Variables returned by this procedure

None.

Variables used by this procedure

Breath	INTEGER variable used for loop counting and indexing into the respiratory data array Resp_var.
F_name	SHORT_STRING variable containing the file name entered by the user.
File_name	SHORT_STRING variable used to store file names while opening both of the files used by this procedure.
File_ASCII	TEXT file descriptor for the file that will store some of the calibration and analysis information. Required for working with ASCII and text files.
File_name_REAL	SHORT_STRING variable used for the file name used to store the analyzed respiratory data.
File_REAL	FILE OF REAL. File descriptor for the file that will store the analyzed data.

Good_name	BOOLEAN variable used by the error trapping procedure. True indicates that all file I/O was accomplished without error. False indicates that an error occurred during file I/O.
I	INTEGER variable used for loop counting and indexing.
MSI_device	SHORT_STRING variable set equal to the volume ID of the mass storage device that contains the file.

CONTENTS - APPENDIX G

DOCUMENTATION FOR THE PROGRAM DISPLAY3.TEXT

APPENDIX G

DOCUMENTATION FOR THE PROGRAM DISPLAY3.TEXT

This appendix contains documentation for the plotting and table generation program DISPLAY3. Analysis techniques permit averaging by number of breaths and sliding windows of time. More than one run can be averaged together for a composite run or plotted on top of each ther for visual comparison.

This documentation is organized with all global variables and constants listed first, followed by the procedures and functions in the order they appear in the listing. Many of the variable, procedure, and function descriptions that appear in this appendix are copied in whole or in part from Riblett (2) and Masters (3).

<u>Global Constants - Program: DISPLAY3</u>

Max_breaths	The maximum number of breaths that may be stored in the simulated three dimensional array Resp_var.
Max_degree_of_fit	The maximum degree of fit that will be required by the curve fitting procedure Curve_fit.
Max_index	The number of respiratory parameters that are stored in the three dimensional array Resp_var.
Max_num_pts_for_fit	The maximum number of points that will be fit by the curve fitting procedure.
Max_runs_pgm	The maximum number of runs that will supported by this program. Used for dimensioning arrays.

Print_file_name — The file name used for temporary storage by the RS-232 printer driver procedure Copy_2_printer.

The following fourteen constants are indexes into the simulated three dimensional array Resp_var. The description concerns what is stored in the variable indexed.

Breath_good_index	Equals 1 if for a good breath, 0 otherwise.
CO_prod_index	CO_2 produced for this breath.
Expr_min_vent_index	The expiratory minute ventilation.
FRC_CO2_prod_index	The alveolar CO_2 produced this breath.
FRC_O2_cons_index	The alveolar O_2 consumed this breath.
FRC_resp_quot_index	The alveolar respiratory quotient.
Insp_min_vent_index	The inspiratory minute ventilation.
O2_cons_index	The volume of O_2 consumed this breath.
Resp_freq_index	The respiratory frequency of this breath.
Resp_quotient_index	The respiratory quotient for this breath.
Time_of_brth_index	The calculated time at the start of this breath.
V_tid_insp_index	The inspiratory title volume for this breath.
V_tid_expr_index	The expiratory title volume for this breath.
Volume_of_lung_index	The end-expiration lung volume.

Global variables - Program: DISPLAY3

Avg_runs	BOOLEAN variable. True if several runs are to be averaged together with a standard deviation calculated. Normally False.
Avg_type	SHORT_STRING variable. If equal to "AVERAGE" then average by number of breaths. If equal to "WINDOW" then average by time window. See also W_period, W_width, and No_b_avg.
Char_height	REAL variable equal to the height that characters are to be drawn with the GTEXT graphics procedure.
Char_size	REAL variable equal to the percent of the screen size that each graphic character should be drawn. Used to form the variables Char_height and Char_width. For example, if the plotting surface is to be 40 characters wide set Char_size to 1/40 or 0.025.
Char_width	REAL variable equal to the width hat characters are to be drawn with the GTEXT graphics procedure.
Comment	LONG_STRING ARRAY containing the comments entered by the user when the binary data was analyzed by the program ANALYSIS. One comment for each run loaded.
Corr_flag	BOOLEAN ARRAY containing True if temperature and viscosity corrections were made for this run. False otherwise. One boolean value for each run loaded.
Date	BOOLEAN ARRAY containing the date entered by the user when the binary data was analyzed by the program ANALYSIS. One date for each run loaded.
Degree	INTEGER variable equal to the degree of fit that the polynomial least-squares curve fitting procedure should use.
End_time	REAL variable equal to the time in seconds that windowing procedures should stop at. See also Start_time.
Exit_program	BOOLEAN variable. True if the menu selection flags a return to the main menu. Normally False.
First_time	BOOLEAN variable. True the firs time that the menu is displayed. Set to False when analyzed

data is loaded.

FRC_flag	BOOLEAN ARRAY containing True if FRC calculations were made for this run. False otherwise. One boolean value for each run loaded. Normally true.
Heading	STRING[40] ARRAY containing all of the headings that can be printed at the top of plots. There are currently twelve headings defined.
I	INTEGER variable used for array indexing and loop counting.
Listing	TEXT file required to print data to the listing device. Currently the listing device is the thermal printer.
Max_runs	INTEGER variable equal to the maximum number of runs that can be loaded into memory. This is determined by the total amount of memory and must be less than the global constant Max_runs_pgm.
Menu_choice	INTEGER variable equal to the menu choice entered by the user.
More_than_one	BOOLEAN variable. True if more that one run may be loaded by the retrieve analyzed data procedure Rt_anal_data.
No_b_avg	INTEGER variable. If Avg_type equals "AVERAGE" then No_b_avg equals the number of breaths that will be averaged to form one data point. See also Avg_type
Num_breaths	INTEGER ARRAY containing the number of breaths stored for each run in memory.
Num_breaths_max	INTEGER variable equal to the maximum number of breaths that can be stored in memory.
Num_string	SHORT_STRING variable used for numeric reads. HP Pascal crashes if a character is entered during a numeric read. See also the procedure Value which converts a string to a real value.
Omit_bad	BOOLEAN variable. True if the user wishes to omit bad breaths. False otherwise. A bad breath is defined as being less than 0.4 liters of inspiratory or expiratory gas.
Plots	INTEGER variable equal to the number of plots that

	are to be made if more than one run is to be used for each plot.
Q	SHORT_STRING variable used for all yes/no questions. See also the procedure Ask_y_or_n in the Utilities description.
Resp_var_array	ARRAY of POINTERS into the three dimensional array used to store the results of the analyzed data.
Result_type	SHORT_STRING variable that indicates the menu choice with a string. Possible values are: PLOT, PRINT, PRINT_AVG, VAR_VAR, CURVE_FIT, CURVE_FIT_AVG, and PLOT_AVG.
RQ_type	INTEGER variable indicating which respiratory quotient value should be printed. 1=RQ, 2=average CO_2/ average O_2 for that breath, 3=FRC RQ.
Runs	INTEGER variable equal to the number of analyzed runs that are loaded into memory.
Selection	INTEGER variable equal to the menu selection.
Skip_axes	BOOLEAN variable. True if the user does not wish to have the axes drawn. Normally False.
Start_time	REAL variable equal to the time in seconds that windowing procedures should start averaging data. See also End_time.
Subject_name	SHORT_STRING ARRAY containing the subject's name. One value for each run in memory.
W_period	REAL variable equal to the sliding window's period in seconds. See also Avg_type.
W_width	REAL variable equal tho the sliding window's width in seconds. See also Avg_type.

Function: **Resp_var**

This function emulates reading from a three dimensional array of type REAL by using many single dimensional dynamic arrays. This technique must be used in Pascal because there is not enough global storage for one three dimensional array the size that this one needs to be. This is not an elegant solution to the problem, however it works and is accurate. See also the procedure Resp_var_write.

<u>Variables passed to this function</u>

Breath — INTEGER value equal to the breath number that is desired.

Index — INTEGER value equal to the respiratory parameter that is desired.

Run — INTEGER value equal to the run number that is desired. No error checking is done to ensure that this is a valid run.

<u>Variables returned by this function</u>

None.

<u>Variable used by this function</u>

None.

Procedure: Resp_var_write

This procedure emulates writing to a three dimensional array of type REAL by using many single dimensional dynamic arrays. This technique must be used in Pascal because there is not enough global storage for one three dimensional array the size that this one needs to be. See also the function Resp_var.

Variables passed to this procedure

Breath	INTEGER value equal to the breath number that is desired.
Index	INTEGER value equal to the desired respiratory parameter.
Run	INTEGER value equal to the run number that is desired. No error checking is done to ensure that this is a valid run.
Value	REAL variable equal to the data that is to be stored in the three dimensional array Resp_var.

Variables returned by this procedure

None.

Variables used by this procedure

None.

Function: Log

This function of type REAL returns the base 10 logarithm of the argument passed to the function. HP Pascal only supplies a natural logarithm function.

Variables passed to this function

X — REAL value that is the argument of the base ten logarithm function.

Variables returned by this function

None.

Variables used by this function

None.

Procedure: Curve_fit

This procedure performs an Nth order polynomial least squares curve fit to a set of input vectors. It is not recommended that a fit greater than 6th order be used by this procedure. The matrix inversion technique uses LU decomposition and the corresponding procedures (Ludcmp and Lubksb) come directly from the text Numerical Recipes The Art of Scientific Computing, Press, Flannery, Teukolsky, Vetterling, p. 31-38, 683-684.

Variables passed to the procedure

Degree	INTEGER variable equal to the degree of fit.
Num_pts	INTEGER variable equal to the number of (X,Y) points that are to be used in the fit.
X_vector	GLNARRAY array variable containing the X coordinates.
Y_vector	GLNARRAY array variable containing the Y coordinates.

Variables returned by the procedure

Poly	GLNRESULT array variable containing the coefficients. Poly[1] contains the zero order coefficient.

Variables used by the procedure

A	GLNSQUARE. Square matrix used for intermediate calculations during the least-squares curve fitting process.
A_tran	GLNPBYNP This is a rectangular matrix containing the x_vector and the required powers of the x_vector.

B	GLNRESULT Matrix product: B=A_tran*Y_vector where Y_vector is one of the input vectors.
C	GLNSQUARE Inverse of the A matrix.
Col	INTEGER variable use for array indexing.
D	INTEGER variable required by the LU decomposition and back-substitution procedures.
Deg	INTEGER variable used for array indexing.
I	INTEGER variable used for array indexing.
Indx	INTEGER variable used for array indexing.
J	INTEGER variable used for array indexing.
N	INTEGER variable equal to the degree of fit plus one.
Row	INTEGER variable used for array indexing.
Sum	REAL variable used for intermediate calculations during matrix multiplication.

Procedure: Copy_2_printer

This procedure copies a text file from disk to the DecWriter II RS-232 printer. This procedure is required because HP Pascal only allows one system printer to be defined. The thermal printer is defined as the system printer so it can be used for graphic screen dumps. An XON/XOFF handshake protocol is implemented because the printer's buffer overflows during continuous printing. Although the HP 9826 supports hardware handshaking, the DecWriter's interface does not.

Variables passed to this procedure

None.

Variables returned from this procedure

None.

Constants defined by this procedure

Prt_device	The hardware HPIB select code for the printer that will receive the outgoing data. The RS-232 interface has select code 9 while the thermal printer is select code 701.
XOFF	The standard ASCII character for Stop-transmit.
XON	The standard ASCII character for Start-transmit.

Variables used by this procedure

C	CHARACTER variable used to store incoming characters from the RS-232 serial printer.
Line	STRING[200] variable used for loading the text lines that will be printed on the serial printer.
T	TEXT file descriptor required for text file I/O.

Procedure: Avg_breath

This procedure averages a requested number of breaths from the analysis routine. The calling procedure tells which run to use, which breath to start with, and the number of breaths to be used in the average. Bad breaths may be omitted in the average.

Variables passed to this procedure

Br_pntr	INTEGER value equal to the breath number that will be used for the first breath in the average.
No_b_avg	INTEGER value equal to the number of breaths that will be averaged.
Run_num	INTEGER value equal to the run number that will be averaged.
Variable	INTEGER value equal to the respiratory parameter that will be averaged.

Variables returned by this procedure

Average	REAL variable equal to the calculated averaged for the set of breaths.
Num_bad_breaths	INTEGER variable equal to the number of bad breaths that were omitted while averaging.
Time	REAL variable equal to the average time between the first and last breaths averaged.

Variables used by this procedure

J	INTEGER variable used for array indexing.
Pointer	INTEGER variable used as a pointer into the respiratory data array.

Procedure: Avg_window

This procedure calculates an average of the breaths occurring in a time window. The calling procedure must supply the center time and the number of seconds wide for the window. Bad breaths may be omitted during the averaging.

Variables passed to the procedure

Br_pntr	INTEGER variable pointing to a recommended first breath for the window. The windowing procedure will verify that this is the proper starting breath.
Run_num	INTEGER value indicating the run number to use.
Variable	INTEGER value equal to the respiratory parameter to use.
W_center	REAL variable equal to the time in seconds equal to the center of the time window.
W_width	REAL variable equal to the window width in seconds.

Variables returned by the procedure

Average	REAL variable equal to the average of the breaths found in the window.
Br_in_w_flag	BOOLEAN variable. True if any breaths were found in the time window. False otherwise.
Br_pntr	INTEGER variable equal to the breath number of the last breath in the window.

Variables used in the procedure

Left_edge_w	REAL variable set equal to the time in seconds of the start of the sliding time window.
Fstbr_in_w_fnd	BOOLEAN variable. Initially set False. Set True when the breath pointer variable, Br_pntr, points to the first breath in the time window.
Right_edge_w	REAL variable set equal to the time in seconds of

the ending of the sliding time window.

Num_b_in_window	INTEGER variable set equal the number of breaths found in the time window. Used to calculate the average of the window.

Procedure: Print_results_printer

This procedure controls the averaging of the data when the output is to be directed to the printer. The print file is directed to a disk file for later printing by the procedure Copy_2_printer.

Variables passed to the procedure

Avg_type	SHORT_STRING variable used to determine the averaging technique to use. If equal to 'WINDOW' then use time windowing. If equal to 'AVERAGE' then average a constant number of breaths.

Variables returned by the procedure

None.

Variables used by the procedure

Average	REAL variable set equal to the average of the respiratory parameter for the current time or breath window.
Br_in_w_flag	BOOLEAN variable. True if there was a breath in the current time window. False otherwise.
Br_index	INTEGER variable used as a temporary pointer into the respiratory data array.
Br_pntr	INTEGER variable used as the main pointer into the respiratory data array.
I	INTEGER variable used for loop counting.
Num_bad_breaths	INTEGER variable equal to the number of bad breaths encountered while averaging a time window.
Run_num	INTEGER variable used for loop counting and array indexing.
Some_rq_too_big	BOOLEAN variable. Normally False. Set True if a respirator quotient was calculated to be greater than or equal to 100.
Sum	REAL variable used as a summing register during

calculation of averages.

T	TEXT file used for temporary storage of the data to be printed.
Time	REAL variable set equal to the average time of the breaths used for the generation of the variable Average.
Values	ARRAY of REAL variables equal to the respiratory variables that are to be printed on this line.
W_center	REAL variable equal to the time in seconds that represents the center of the sliding time window.
X	INTEGER variable used for loop counting.

Procedure: Print_line

Sub proc of: Print_results_printer

This procedure prints one line of averaged data to the print file. The output format is controlled by this procedure.

Variables passed to this procedure

None.

Variables returned by this procedure

Some_rq_to_big	BOOLEAN variable. If a respiratory quotient is too large to fit the print format then this variable is set True. Otherwise this variable is not modified.

Variables used by this procedure

None.

Procedure: Print_averages

This procedure prints the averages and standard deviations of the respiratory parameters for several runs. Each run is averaged by the windowing technique, then the window average of each run is averaged to form an overall average for all runs within that time window.

Variables passed to this procedure

None.

Variables returned by this procedure

None

Variables used by this procedure

Amount	REAL variable used as a summation resister for calculations of averages.
Average	REAL variable equal to the average returned from the window averaging procedure.
Avg	ARRAY of REAL used to store the variable Average for all of the runs.
Br_in_w_flag	BOOLEAN variable. True if there was a breath in the current time window. False otherwise.
Br_index	INTEGER variable used as a temporary pointer into the respiratory data array.
Br_pntr	INTEGER variable used as the main pointer into the respiratory data array.
Diff	REAL variable use for standard deviation calculation.
I	INTEGER variable used for loop counting.
Num_bad_breaths	INTEGER variable equal to the number of bad breaths encountered while averaging a time window.
Quot	SHORT_STRING variable set equal to different

	strings depending on the method of calculating the respiratory quotient.
Resp	SHORT_STRING variable set equal to different strings depending on the method of calculating the respiratory quotient.
Some_rq_to_big	BOOLEAN variable. Normally False. Set True if a respirator quotient was calculated to be greater than or equal to 100.
St_dev	ARRAY of REAL variables equal to the standard deviation of the respiratory parameters. These values will be printed.
T	TEXT file used for temporary storage of the data to be printed.
Total	REAL variable use for standard deviation calculation.
Values	ARRAY of REAL variables equal the respiratory variables that will be printed.
W_center	REAL variable equal to the time in seconds that represents the center of the sliding time window.
X	INTEGER variable used for loop counting for the run number.
Y	INTEGER variable used for loop counting for the run number.

Procedure: Max_min

As the name implies, this procedure finds the maximum and minimum of a window of respiratory data. Either number of breaths or time windows may be used in the maximum/minimum search. This procedure is called when the user chooses to use default axis ranges.

Variables passed to the procedure

Avg_type	SHORT_STRING variable used to determine the averaging technique to use. If equal to 'WINDOW' then use time windowing. If equal to 'AVERAGE' then average a constant number of breaths.
Run_num	INTEGER value equal to the run number to use.
Variable	INTEGER value equal to the respiratory parameter to use for the search.
Max	REAL variable equal to the maximum value located in the search.
Min	REAL variable equal to the minimum value located in the search.

Variables used in the procedure

Average	REAL variable equal to the average of the respiratory parameter as returned from either the Avg_breath or Avg_window procedures. Also used as the average of several runs if more than one run is being processed.
Br_in_w_flag	BOOLEAN variable. True if there was a breath in the current time window. False otherwise.
Br_pntr	INTEGER variable used as the main pointer into the respiratory data array.
K	INTEGER variable used for loop counting.
Num_bad_breaths	INTEGER variable equal to the number of bad breaths encountered while averaging a time window.

Temp_average	REAL variable equal to the average of the respiratory parameter as returned from either the Avg_breath or Avg_window procedures. Used for generation of the average of more than one run.
Time	REAL variable equal to the average time if breath averaging technique is used. Not used in any calculations in this procedure.
W_center	REAL variable equal to the time in seconds that represents the center of the sliding time window.

Procedure: Set_pen_speed

This procedure is used to set the plotter's pen speed to a known value. By default, the plotter runs at full blast, which is 30 cm/sec maximum. A typical maximum pen speed for good plots is 5 cm/sec.

This procedure is copied from page 85 of the HP Pascal 3.0 Graphics Procedures Manual.

Variables passed to the procedure

Speed	INTEGER variable equal to the desired maximum pen speed in cm/sec. A maximum value of 30 cm/sec is allowed by the current eight-pen vector plotter.

Variables returned by the procedure

None.

Procedure: Make_axes

This procedure draws and labels a set of axes on the plotting device. It is used if either one or two plots are to be made on the same piece of paper.

Variables passed to the procedure

Max	REAL value equal to the maximum Y value for the axes.
Min	REAL value equal to the minimum Y value for the axes.
One_or_two	INTEGER value. Equal to one if only one plot set of axes are to be drawn. Equal to two of two sets of axes are to be drawn.
Skip_axes	BOOLEAN variable. True if the procedure should not draw the axes. False if the axes should be drawn and labeled.
Variable	INTEGER variable equal to the respiratory parameter that is to be plotted. Required for titling the plot.

Variables returned by the procedure

None.

Variables used by the procedure

Aspect	REAL variable used to alter the height to width ration of characters to be plotted using the GTEXT procedure.
Bottom	REAL variable containing the virtual coordinates for the bottom edge of the plotting window.
I, J	INTEGER variables used for general purpose indexing.
Left	REAL variable containing the virtual coordinates for the left most edge of the plotting window.
Num_string	SHORT_STRING variable used for numeric reads. See also Num_string description in Global variables.

Num_ticks	INTEGER variable equal to the number of tick marks that are to be drawn on each of the axes.
Plot_label	STRING[100] variable used for string storage before plotting text information with the GTEXT procedure.
Right	REAL variable containing the virtual coordinates for the right most edge of the plotting window.
String_length	INTEGER variable containing the length of the string to be plotted using the GTEXT procedure. Used for centering the string.
Top	REAL variable containing the virtual coordinates for the top edge of the plotting window.
X, Y	REAL variables used for general purpose axes values.

Procedure: Make_axes2

This procedure draws and labels a set of axes for the variable vs. variable plotting option.

Variables passed to the procedure

Fig1	INTEGER variable equal to the respiratory parameter that is to be plotted in the dependent (Y) axis.
Fig2	INTEGER variable equal to the respiratory parameter that is to be plotted on the independent (X) axis.
Max1	INTEGER variable equal to the maximum axis value for the dependent axis.
Max2	INTEGER variable equal to the maximum axis value for the independent axis.
Min1	INTEGER variable equal to the minimum axis value for the dependent axis.
Min2	INTEGER variable equal to the minimum axis value for the independent axis.

Variables returned by the procedure

None.

Variables used by the procedure

Bottom	REAL variable containing the virtual coordinates for the bottom edge of the plotting window.
Final_length	INTEGER variable equal to the length of the text string to be plotted. Used for centering.
I, J	INTEGER variables used for general purpose indexing.
Left	REAL variable containing the virtual coordinates for the left most edge of the plotting window.
Num_string	SHORT_STRING variable used for numeric reads. See also Num_string description in Global variables.
Plot	INTEGER variable used for indexing while labeling the axes.

Plot_line	LONG_STRING variable used for string storage before plotting text information with the GTEXT procedure.
Right	REAL variable containing the virtual coordinates for the right most edge of the plotting window.
Top	REAL variable containing the virtual coordinates for the top edge of the plotting window.
Variable	INTEGER variable equal to the respiratory parameter to be plotted.
X, Y	REAL variables used for general purpose axes values.
Xtick, Ytick	REAL variables used for general purpose axes values.

Procedure: Plot_values

This procedure plots the analyzed respiratory data to either the CRT or the plotter. This procedure assumes that the virtual window and plotting surface have been initialized and are ready for axis generation and plotting.

Variables passed to the procedure

Avg_type	SHORT_STRING variable used to determine the averaging technique to use. If equal to 'WINDOW' then use time windowing. If equal to 'AVERAGE' then average a constant number of breaths.
Run_num	INTEGER value equal to the run number to use.
Variable	INTEGER value equal to the respiratory parameter to plot.
Max	REAL variable equal to the maximum Y axis value.
Min	REAL variable equal to the minimum Y axis value.

Variables returned by the procedure

None.

Variables used by the procedure

Avg	ARRAY of REAL variables used to store the averages returned by the window averaging procedure when the standard deviation is to be calculated.
Average	REAL variable equal to the average of the respiratory parameter as returned from either the Avg_breath or Avg_window procedures.
Br_in_w_flag	BOOLEAN variable. True if there was a breath in the current time window. False otherwise.
Br_pntr	INTEGER variable used as the main pointer into the respiratory data array.
Diff	REAL variable used in standard deviation calculation.

I	INTEGER variable used for loop counting.
K	INTEGER variable set equal to the current run number when more than one run is to be averaged for each plot.
L	INTEGER variable used for loop counting.
Num_bad_breaths	INTEGER variable equal to the number of bad breaths encountered while averaging a time window.
St_dev	REAL variable equal to the standard deviation for the averaged points.
T	INTEGER variable used for looping counting.
Time	REAL variable equal to the average time of the breaths averaged by the Avg_breath procedure.
Total	REAL variable used in standard deviation calculation.
Total_avg	REAL variable used as a summation register for calculation of averages.
W	INTEGER variable used for a loop endpoint when more than one run is to be averaged for each plot.
W_center	REAL variable equal to the time in seconds that represents the center of the sliding time window.
X	INTEGER variable used for the starting index for the averaging loop when more than one run is to be averaged for each plot.
Y	INTEGER variable used for loop counting when generating standard deviations.

Procedure: Plot_control

Plot_control contains the user interface for the plotting routines. The user enters the respiratory parameters that are to be plotted along with the minimum and maximum axis values. Default axis limits may also be selected. Plot_control does all of the plotting if curve fitting is selected. Otherwise plot_control calls the procedure Plot_values to have points plotted.

<u>Variables passed to the procedure</u>

None.

<u>Variables returned by the procedure</u>

None.

<u>Variables used by the procedure</u>

Abor — BOOLEAN variable. Normally False. Set true if a mathematical problem develops during curve fitting.

Amount — REAL variable used as a summation register for calculating the averages is curve fitting is to be used.

Average — REAL variable equal to the calculated average of a window of breaths. May be returned from the procedure Avg_window or calculated in this procedure.

Avg_bad_comnt — STRING[100] variable that contains a text string to be printed at the bottom of finished plots. The string will contain information on the number of breaths averaged, window sized, and if bad breaths were omitted.

Brth_in_w_flag — BOOLEAN variable. True if there was a breath in the current time window. False otherwise.

Br_index — INTEGER variable used as a temporary pointer into

	the respiratory data array.
Br_pntr	INTEGER variable used as the main pointer into the respiratory data array.
Error_return	INTEGER variable used by the HP Pascal graphics procedures. Non-zero if a graphics error occurs during initialization of the graphics device.
F	REAL variable representing the statistical F parameter. Not currently generated.
Fig	ARRAY[1..2] OF INTEGER variables. Contains the menu choices of the respiratory variables to be plotted. Menu choice number 1 is O_2 consumed, number 2 is CO_2 produced, etc.
I	INTEGER variable used for loop counting.
II	INTEGER variable used for loop counting.
K	INTEGER variable equaling the run number currently being plotted. Only used for curve fitting data.
Last_run	INTEGER variable set equal to one if curve fitting an averaged curve is to be used. Otherwise set equal to the number of runs loaded into memory.
Max	ARRAY OF REAL variables containing either the user's entry or the calculated maximums of the figures to be plotted.
Max_string	ARRAY[1..2] OF SHORT_STRING variables containing the user's entry of the maximum plot values for figures one and two.
Min	ARRAY OF REAL variables containing either the user's entry or the calculated minimums of the figures to be plotted.
Min_string	ARRAY OF SHORT_STRING variables containing the user's entry of the minimum plot values for figures one and two.
Num_bad_breaths	INTEGER variable equal to the number of bad breaths encountered while averaging a time window.
Num_string	SHORT_STRING variable used for numeric reads. See also the Num_string description in Global variables.

One_or_two	INTEGER variable equal to the number of axes that are to be generated. May only equal one or two.
Pen_list	ARRAY of INTEGER variables. Each element contains the pen number that will be used for the associated run that is plotted.
Plotting_device	INTEGER variable equal to 3 if the CRT is the plotting device to use. Equal to 705 if the plotter is the desired plotting device.
Plot_height	REAL variable equal to the fraction of the plotting device's height that will be used for setting the virtual plotting window. If two plots are to be made then this must be less than one half. See also Plot_top.
Plot_line	LONG_STRING variable containing text that will be plotted using the GTEXT graphics procedure.
Plot_sym	ARRAY OF SHORT_STRING variables equal to a set of symbols that may be plotted.
Plot_top	REAL variable equal to the edge of the plotting surface that will become the new virtual plotting window. A value of zero indicates the bottom edge while one indicates the top. See also Plot_height.
Q	SHORT_STRING variable used for all yes/no questions. See also the procedure Ask_y_or_n in the Utilities description.
Result_poly	GLRESULT variable containing the result polynomial coefficients returned from the Curve_fit procedure.
Ru	INTEGER variable used for counting the number of invalid points if curve fitting of averaged data is requested.
S	INTEGER variable used for loop counting if curve fitting of averaged data is requested.
Step_size	REAL variable used for generating the numeric axes labels.
Temp_max	REAL variable used for temporary storage of maximums.
Temp_min	REAL variable used for temporary storage of

minimums.

Time	REAL variable equal to the average time of the breaths averaged by the procedure Avg_breaths.
Total	INTEGER variable equal to the total number of points that are to be fit using the curve fitting procedure.
W_center	REAL variable equal to the time in seconds that represents the center of the sliding time window.
X	GINARRAY Array variable containing the X values of the data to be curve fitted. See also Y.
X_aver	REAL variable containing the average returned from the window averaging procedure. Used while calculating the averages of more than one run. See also Y_aver.
X_val	REAL variable used for temporary storage of x coordinates while drawing the curve calculated by the curve fitting procedure. Also used for temporary storage of x coordinates while setting up the points to be fit by the curve fitting procedure. See also Y_val.
Y	GINARRAY Array variable containing the Y values of the data to be curve fitted. See also X.
Y_aver	REAL variable containing the average returned from the window averaging procedure. Used while calculating the averages of more than one run. See also X_aver.
Y_val	REAL variable used for temporary storage of y coordinates while drawing the curve calculated by the curve fitting procedure. Also used for temporary storage of y coordinates while setting up the points to be fit by the curve fitting procedure. See also X_val.

Procedure: Show_help

This procedure copies several screens of text from the disk file 'HARD1:HELP_FILE.TEXT' to the CRT. The screens are paused at convenient intervals so the user is not lost. The help screens give a short description of all of the menu commands and averaging techniques for the DISPLAY program.

Variables passed to the procedure

None.

Variables returned by the procedure

None.

Variables used by the procedure

Count	INTEGER variable used to count the number of lines printed on the screen since the last form feed. Used to keep text from scrolling off the screen before it has been read.
Line	LONG_STRING variable used to hold the text that is read from the disk to be printed to the screen.
T_file	TEXT file descriptor required for Pascal to read a text file from disk.

Procedure: Window_period

This procedure allows the operator to enter the desired window period, window width, start time, and stop time for the sliding time window averaging technique. The user may select the default entry by pressing only ENTER for the desired value. All four variables have units of seconds.

Variables passed to this procedure

Period	REAL value equal to the default (previous) value for the window period.
Start	REAL value equal to the default (previous) value for the window start time.
Stop	REAL value equal to the default (previous) value for the window end time.
Width	REAL value equal to the default (previous) value for the window width.

Variables returned by this procedure

Period	REAL value equal to the entered window period.
Start	REAL value equal to the entered start time.
Stop	REAL value equal to the entered ending time.
Width	REAL value equal to the entered window width.

Variables used by this procedure

Num_string	SHORT_STRING variable used for numeric reads.

Procedure: Combine_breaths

This procedure allows the operator to select the method of breath averaging. Valid choices are plot the individual breaths (no averaging), average a constant number of breaths, and average using a sliding time window.

Variables passed to this procedure.

None.

Variables returned by this procedure.

Avg_runs	BOOLEAN variable. Set True if the user wishes to generate standard deviations along with averaged runs.
Avg_type	SHORT_STRING variable set equal to 'WINDOW' if the sliding time window technique is to be used. Otherwise set equal to 'AVERAGE' to indicate averaging a set number of breaths.
No_b_avg	INTEGER variable. If averaging a set number of breaths is selected then this variable equals the number of breaths that will be averaged. If the individual breaths are to be used then this variable equals one.

Variables used by this procedure

Choice	INTEGER variable used to store the menu choice. Valid values are either 1, 2, or 3.
K	INTEGER variable used for loop counting.
Num_string	SHORT_STRING variable used for numeric reads.

Function: Omit_bad_breaths

This BOOLEAN function asks the user if they want to omit bad breaths from the averaging routines. A bad breath is defined to be less than 0.4 liters of inspirate or expirate volume. This function returns True if bad breaths are to be omitted. Otherwise False is returned.

Variables passed to this procedure

None.

Variables returned by this procedure

None.

Variables used by this procedure

Q	SHORT_STRING variable used for all yes/no questions. See also the procedure Ask_y_or_n in the Utilities description.

Procedure: Rt_anal_data

This procedure loads previously analyzed data from the disk and stores the data in the simulated three dimensional array Resp_var. More than one set of run may be loaded into memory for calculations. The data files are created by the program ANALYSIS and are in the same format that ANALYSIS stores analyzed data. See the ANALYSIS documentation for a description of this format.

Variables passed to this procedure

First_time	BOOLEAN variable. If True then Rt_anal_data assumes that there is no data currently in memory and does not bother to ask the user if new data should be loaded.
More_than_one	BOOLEAN variable. If True than more than one run may be loaded into memory. If False then only one data set may be displayed for the current function.

Variables returned by this procedure

First_time	BOOLEAN variable always set False.

Variables used by this procedure

A	INTEGER variable used for array indexing.
Data_value	REAL variable used for temporary storage of all data values as they are loaded from the disk.
F_name	SHORT_STRING variable containing the file name as entered by the user.
File_ASCII	TEXT file descriptor required for accessing text and ASCII files using HP Pascal.
File_name	SHORT_STRING variable containing the file name as entered by the user plus the mass storage volume ID, name prefix, and file type for both the ASCII file and the file of REAL values.

File_REAL	FILE OF REAL descriptor required for accessing data files that contain only REAL values.
Good_name	BOOLEAN variable used by the error trapping procedure. True indicates that all file I/O was accomplished without error. False indicates that an error occurred during file I/O.
I, J, K	INTEGER variables used for loop counting.
Load_some_data	BOOLEAN variable. Set True if the user wishes to load a new set of data. Otherwise set False.
MSI_device	SHORT_STRING variable set equal to the volume ID of the mass storage device that contains the desired file.
Num_string	SHORT_STRING variable used for numeric reads. See also the Num_string description in Global variables.
S	INTEGER variable used as a multiplying constant when indexing into arrays.

CONTENTS - APPENDIX H

DOCUMENTATION FOR THE PROGRAM EDIT_FRC.TEXT

APPENDIX H

DOCUMENTATION FOR THE PROGRAM EDIT_FRC.TEXT

This appendix contains documentation for the utility program that is used to enter default age, weight, and height values for the test subjects. The analysis program requires subject information in order to make an estimate of the default resting lung volume, also known as the functional residual capacity or FRC, of the subject. FRC estimates are required by ANALYSIS3 to calculate the alveolar concentrations and volume of CO_2 and O_2.

The default parameters are stored in a disk file under the file name HARD0:FRC_INFO.TEXT. The file name may be changed in this program by editing the constant File_name. If the file name is changed in this program then the program ANALYSIS must by revised also. Edit the statement 'RESET(F_text, 'HARD0:FRC_INFO.TEXT');' in the procedure Load_FRC_subject_info of the analysis program if the file name is changed.

The format of the default subject information is described in Figure M.1. The file is stored as a text file so that either this editing program or the system editor may be used to edit the default information. Currently the program supports a maximum of 30 subjects. The maximum may be changed by editing the constant Max_subjects.

Line number	Contents of the line
1	FORMAT OF THIS DATA FILE:
2	NAME
3	AGE (yrs) HEIGHT (inches) WEIGHT (lbs)
4	START
5	$name_1$
6	age_1 $height_1$ $weight_1$
7	$name_2$
8	age_2 $height_2$ $weight_2$
2*n+4	age_n $height_n$ $weight_n$
2*n+5	END
2*n+6	0 0 0

Notes: n=subject's number
There may be any number of comment lines prior to the line containing "START".

Figure H.1. Format of the file HARDO:FRC_INFO.TEXT.

Global Constants - Program: EDIT_FRC

File_name — The file name containing the default subject information. Currently equal to "HARDO:FRC_INFO.TEXT".

Max_subjects — The maximum number of subjects that may be stored in the default file. Currently equal to thirty.

Global Variables - Program: EDIT_FRC

Age — ARRAY OF INTEGER variables. Each index contains the age in years of the indexed subject.

Count — INTEGER variable equal to the number of subjects contained in the default file.

Delete_flag — INTEGER variable equal to the subject number of the subject that is to be removed from the file of defaults.

Exit_program — BOOLEAN variable. True if the user chooses the exit program menu selection. Normally False.

Height — ARRAY OF INTEGER variables. Each index contains the height in inches of the indexed subject.

I — INTEGER variable used for loop counting and array

indexing.

Listing	TEXT file descriptor. Required to use the system printer.
Menu_choice	INTEGER variable equal to the menu choice entered by the user. Error checking is done to protect against invalid entries.
Name	ARRAY OF SHORT_STRING variables. Each index contains the name of the subject.
Num_string	SHORT_STRING variable used for entry of numeric data. The function Value, described in the Utilities appendix, converts the string into a REAL value.
Weight	ARRAY OF INTEGER variables. Each index contains the weight in pounds of the indexed subject.

Procedure: Load_FRC_info

This procedure loads the current default subject information from the disk and stores the data in the arrays Name, Age, Weight, and Height. Any leading comments that may be stored in the file are ignored.

Variables passed to the procedure

None.

Variables returned by the procedure

Count	INTEGER variable equal to the number of subjects loaded from the disk file.

Variables used by this procedure

Dummy	STRING[100] used for temporary storage while searching for the start of the subject information section of the file.
F	TEXT file descriptor required by HP Pascal to read from a text file.

Procedure: Save_FRC_info

This procedure saves the default information back to the disk after modifications are made. The old file is deleted before the new file is written.

Variables passed to this procedure

Count	INTEGER variable equal to the number of subjects that are stored in the name, age, height, and weight arrays.
Delete_flag	INTEGER variable equal to the subject number of the subject that is to be removed from the file of defaults. If Delete_flag=0 or Delete_flag>Count then no subject will be removed from the file.

Variables returned by this procedure

None.

Variables used by this procedure

F	TEXT file descriptor required by HP Pascal to write to text files.
I	INTEGER variable used for loop counting and array indexing.

CONTENTS - APPENDIX I

DOCUMENTATION FOR THE PROGRAM SET_DATE.TEXT

APPENDIX I

DOCUMENTATION FOR THE PROGRAM SET_DATE.TEXT

This appendix contains documentation for the utility program that is used to set the current date within the memory of the computer. The date is useful because whenever a program or file is modified the current date and time is also recorded with the file. The system filer can then be used to examine disks for old files that can be deleted.

The auto-start stream file for the operating system automatically executes this program each time that HP Pascal 3.0 is booted. This program can also be executed from the main menu of the computerized respiratory analysis software.

Refer to the System Devices section of the HP Pascal 3.0 Procedures Reference Manual for further information and examples on this technique.

Global Variables - Program: SET_DATE

Date	DATEREC variable. The variable type is defined by the sysdevs operating system import module. This variable is used to store the system date while communicating with the operating system.
Day	INTEGER variable equal to the day of the month. 1=first, 2=second, etc.

Entry	SHORT_STRING variable equal to the ASCII text string that is entered by the user for the new date.
Good_date	BOOLEAN variable. Set True if the user entered a valid date. Set False if the user entered an invalid date. The program will not exit until the user enters a valid date.
Month	INTEGER variable equal to the month. 1=January, 2=February, etc.
Num_string	SHORT_STRING variable used for entry of numeric data by the user. HP Pascal crashes if the user presses a non-numeric key when Pascal is expecting a numeric entry. The procedure Value, described in the Utilities appendix, converts a string into an equivalent REAL value.
Year	INTEGER variable equal to the current year. 85=1985, 86=1986, etc.

CONTENTS - APPENDIX J

DOCUMENTATION FOR THE PROGRAM DISK_SCAN.TEXT

APPENDIX J

DOCUMENTATION FOR THE PROGRAM DISK_SCAN.TEXT

This appendix contains documentation for the utility program that is used to scan disk directories for files. The calibration factors, binary data, and analyzed results all have unique file name prefixes, it is possible to search for the prefixes to identify files. The main document of this thesis describes the file name prefixes in detail.

The output of this program is sent to the system printer. The system printer is currently defined to by the thermal printer.

This program exploits the possibilities of adding characters to the keyboard buffer by using the procedure Send_to_kbd. Send_to_kbd is documented in Appendix K.

Global Variables - Program: DISK_SCAN

I	INTEGER variable used for loop counting and array indexing.
Listing	TEXT file descriptor. Required to print text on the system printer.
Menu_choice	INTEGER variable equal to the menu choice entered by the user. Error checking is performed to ensure a valid entry.
MSI_device	SHORT_STRING variable equal to the volume ID of the mass storage device that will be searched. Returned by the procedure Select_MSI.

Num_string SHORT_STRING variable used for entry of numeric data by the user. HP Pascal crashes if the user presses a non-numeric key when Pascal is expecting a numeric entry. The procedure Value, described in the Utilities appendix, converts a string into an equivalent REAL value.

Work_string KBD_CTRL_STR variable used to store the key strokes that will be stored in the keyboard buffer by the procedure Send_to_kbd.

CONTENTS - APPENDIX K

DOCUMENTATION FOR THE UTILITY FILE UTILITIES.TEXT

APPENDIX K

DOCUMENTATION FOR THE UTILITY FILE UTILITIES.TEXT

All of the Pascal programs included in this documentation share a common set of functions and procedures called the utility file. This appendix documents the nine utilities. The utility file was created so that all of the programs would not need to be modified if there are any changes in the common set of utilites. HP Pascal programs can use the utility file by adding the following statement to their programs after the variable declarations.

```
$INCLUDE 'HARD1:UTILITIES.TEXT'$
```

The command must be left justified in the source file. The volume ID is assumed to be HARD1: and the file name is UTILITIES.TEXT.

S.1 Compiler Switches

Any program that uses the utilities must include the following compiler switches. Compiler switches inform the compiler that the program may include some non-standard Pascal procedures and functions. The switches that are required by the utilities procedures must be located at the beginning of the source program.

$SYSPROG ON$	This switch allows disk file error trapping to occur.
$SWITCH_STRPOS$	This switch changes the order of the arguments used by the HP Pascal function STRPOS. STRPOS is used in the procedure Up_case.
$UCSD$	This switch allows the compiler to recognise procedures and functions defined by UCSD Pascal but not HP Standard Pascal.

S.2 Type Declarations

The utility procedures and functions also require the following two variable type declarations.

Short_string = STRING[25];

This is a string type that is used for most character strings that the user enteres from the keyboard.

Kbd_ctrl_string = Packed array [1..40] of CHAR;

The keyboard control type is used to define the variable that will contain any string of characters that will be sent to the keyboard's buffer using the procedure Send_to_keyboard.

S.3 System Devices

In order for the Send_to_keyboard procedure to work it is necessary to import an additional set of procedures. The extra procedures allow Pascal programs to access resources normally reserved for the operating system. The new resources that may be accessed include control of the hardware interrupts and several useful small procedures. Interrupts that may be accessed are the keyboard and timers, permitting control of the function keys.

Two of the small additions are a tone generator and the ability to read and set the system's date and time through software. Refer to the SYSTEM DEVICES chapter of the Procedure Reference manual for more

information on these options. To include the system devices functions in a program add the following line to the source code:

```
IMPORT SYSDEVS;
```

Procedure: Form_feed

This procedure sends a form feed character to the CRT. The CRT's software driver interprets the form feed as a command to clear the CRT and place the cursor in the top left corner.

Variables passed to this procedure

None.

Variables returned by this procedure

None.

Variables used by this procedure

None.

Procedure: Line_feed

The Line_feed procedure sends a specified number of line feed characters to the CRT. The CRT's software driver responds to the line feed character by moving the cursor down one line. If the bottom line is reached then the display is scrolled up one line.

Variables passed to this procedure

I INTEGER value equal to the number of lines to go down.

Variables returned by this procedure

None.

Variables used by this procedure

J INTEGER variable used for loop counting.

Procedure: Ask_y_or_n

This procedure prompts the user for a yes or no response. The question must be displayed by the calling program. The calling program can define a default answer by passing that value to the procedure.

Variables passed to this procedure

Q	SHORT_STRING variable equal to the default answer. "N" or "n" indicate a default of no, "Y" or "y" indicate a default of yes. Any other string is interpreted as though no default is available.

Variables returned to this procedure

Q	SHORT_STRING variable equal to the answer entered by the user. "Y" indicates a yes answer and "N" indicates an answer of no. Note that both are upper case characters.

Variables used by this procedure

Q_default	SHORT_STRING variable used to temporarily store the default answer to the yes/no question.

Procedure: Up_case

This procedure converts all of the lower case characters contained in a string to upper case. Non-character elements of the string are ignored. This procedure is useful when creating file names because the HP file system requires upper case characters.

Variables passed to this procedure

Source	SHORT_STRING variable equal to the string that may contain lower case characters.

Variables returned by this procedure

Source	SHORT_STRING variable equal to the upper case equivalent of the string passed to the procedure.

Variables used by this procedure

A	ARRAY [1..25] of STRING[1] variables used to store the individual characters of the original string. Each index stores one character of the original string, up to 25 characters long. 25 is used because that is the length of the SHORT_STRING type. After converting a character to upper case it is returned to A[].
I	INTEGER variable used for loop counting.
Length	INTEGER variable equal to the length of the incoming string.
Pos	INTEGER variable equal to the letter in the alphabet that matches the test letter A[I]. a=1, b=2,...z=26. If no match is found then Pos=0. If a match is found then A[I] is replaced with its upper case equivalent.

Function: Value

This function converts a string variable into a REAL variable containing the numeric equivalent of the string. This function is similar to the function VAL(x$) in BASIC.

Variables passed to this function

Source	SHORT_STRING variable containing the string that needs to be converted.

Variables returned by this function

None.

Variables used by this function

Digit	INTEGER variable equal to the value of the source character. "0"=0, "1"=1, ..., "9"=9, "."=10, "-"=11, all others characters=-1.
Divisor	REAL variable equal to the number of digits found after the decimal point.
Fractional	BOOLEAN variable. Set True if the result contains a fractional part. Set False if the result is an integer.
I	INTEGER variable used for loop counting.
Q	STRING[1] variable containing the next source character to process.
Result	REAL variable equal to the calculated value ignoring the decimal point.

Procedure: Recover_error_10

Error code -10 in the HP Pascal operating system indicates an I/O error. All run-time errors can be trapped within the program. This program, however, only checks for the most common disk and/or file operation errors. The errors that are currently checked for are:

Error Number	Description
7	Bad file name
8	No room on volume
9	Volume not found
10	File not found
18	Media is write protected
30	Attempt to read or write past EOF
31	Media not initialized
32	Device not ready or medium absent
34	Media absent

All errors not trapped by this procedure are returned to the Pascal operating system for processing.

Variables passed to this procedure

Code — INTEGER value equal to the system function ErrorCode.

Result — INTEGER equal to the system function IOResult.

Variables returned by this procedure

Good_name — BOOLEAN variable. False if the error code is implemented in this procedure. True otherwise.

Variables used by this procedure

I — INTEGER variable used for loop counting.

OPTIMIZATION OF A COMPUTERIZED
RESPIRATORY MEASUREMENT SYSTEM

by

GARY IRWIN NOYES

B.S., Kansas State University, 1985

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1987

ABSTRACT

A computerized respiratory measurement system has been developed that can monitor the breath-by-breath ventilatory response of an exercising subject. Corrections are made to reflect changes in gas temperature and viscosity. Results can be printed in tables or plotted on a vector plotter. Averaging techniques for filtering the results include combining multiple breaths and sliding time windows.

The analysis software has been enhanced to increase the execution speed, improve measurement accuracy, and simplify the operating procedure. The maximum data collection time has been increased to thirty minutes. The source code is written entirely in Pascal, which permits easy porting to other computer systems. At a sampling rate of fifty Hertz, less than six minutes are required to analyze nine minutes of respiratory data.

System calibration can be performed by two operators in less than thirty minutes. The majority of the calibration time is used to calibrate the nonlinear flow transducer.